Proyecto Fin de Master
Master en Automática, Robótica y Telemática

MODELADO, PROGRAMACION Y SIMULACIÓN DEL ROBOT IRB 120 DE ABB CON ROBOTSTUDIO

Autor: Beatriz Matos Agudo
Tutor: Manuel Gil Ortega Linares

Dep. Ingenieria de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Master
Master de automática, Robótica y Telemática

MODELADO, PROGRAMACION Y SIMULACIÓN DEL ROBOT IRB 120 DE ABB CON ROBOTSTUDIO

Autor:
Beatriz Matos Agudo

Tutor:
Manuel Gil Ortega Linares
Profesor Titular de Universidad

Dep. de Ingenieria de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Master: MODELADO, PROGRAMACION Y SIMULACIÓN DEL ROBOT IRB 120 DE ABB CON ROBOTSTUDIO

Autor: Beatriz Matos Agudo

Tutor: Manuel Gil Ortega Linares

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi familia

A mis amigos

A mi Profesor

Agradecimientos

Como dijo Thomas A. Edison *"Nuestra mayor debilidad reside en rendirnos. La forma más segura de tener éxito es intentarlo una vez más"*. Por ello, GRACIAS a todas aquellas personas que me han enseñado dia a dia que hay que esforzarse y persistir con lo que uno desea; y sobre todo, que no hay que rendirse jamas.

Beatriz Matos Agudo

Sevilla, 2017

Resumen

El presente trabajo describe los aspectos principales de la programación de aplicaciones, para un manipulador robótico de seis grados de libertad de ABB, más concretamente el modelo IRB120, y su simulación en el software específico del fabricante del robot. El manipulador modelado tendrá fines educativos y será utilizado como herramienta de apoyo didáctico en la Escuela Técnica Superior de Ingeniería Industrial en Sevilla.

El objetivo principal de este proyecto es realizar una serie de guías con los conceptos fundamentales de lenguaje de programación específico para este manipulador y el manejo del software de simulación (RobotStudio) que se empleará. Además también se adaptará el guion de prácticas para poder programar mediante la creación de rutinas en lenguaje RAPID. Con ello, el alumno podrá trabajar tanto con la célula real como en la simulada con idea comparar los resultados finales de la simulación de dicho programa con el comportamiento real del sistema.

Abstract

This project describes the main aspects of the application programming for a robot with six degrees of freedom of ABB, the model IRB120, and the simulation in the software specific of the ABB. The modeled manipulator will have educational purposes and will be used as a teaching support tool at the Higher Technical School of Industrial Engineering in Seville.

The main objective of this project is make a series of guides with the fundamental concepts of programming language specific to this manipulator and the management of the simulation software (RobotStudio) to be used. In addition, the practice script will also be adapted to be programmed by creating RAPID language routines. With this, the student will be able to work with both the actual and the simulated cell in order to compare the final results of the simulation with the real system.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

Wobj	Objeto de trabajo
TCP	Punto central de la herramienta
RS-WCS	Sistema de coordenadas mundo en RobotStudio
TCP(R1)	Punto central de la herramienta del robot 1
TCP(R2)	Punto central de la herramienta del robot 2
BF(R1)	Base de coordenadas de la base del sistema de robot 1
BF(R2)	Base de coordenadas de la base del sistema de robot 2
P1	Objetivo de robot 1
P2	Objetivo de robot 2
TF1	Base de coordenadas de la tarea del sistema de robot 1
TF2	Base de coordenadas de la tarea del sistema de robot 2

1 INTRODUCCIÓN A LA ROBÓTICA

Puede que no sea capaz de definirlo, pero sé cuando veo uno

-Joshep Engelberg -

1.1. Historia de la robótica

El robot es, sin duda, uno de los inventos más populares del siglo XX. La robótica representa la confluencia de varias materias: mecánica, electricidad, electrónica, automática e informática, que hacen que el estudio de un robot resulte enormemente atractivo para especialistas.

Los primeros dispositivos que responden a lo que hoy se conoce como robot, no adoptaron inicialmente esta denominación, dichos dispositivos fueron los manipuladores teleoperados. En 1948 R. C. Goertz del *Argonne National Laboratory* desarrolló el primer sistema de telemanipulación que consistía en un dispositivo mecánico maestro-esclavo, con el objetivo de manejar elementos radioactivos sin riesgo para el operador. En 1054 hizo uso de la tecnología electrónica y del servo control sustituyendo la transmisión mecánica por otra eléctrica, desarrollando así el primer sistema de manipulación con servocontrol bilateral.

La sustitución del operador por un programa de ordenador que controlase los movimientos del manipulador dio paso al concepto de robot, impulsado por la necesidad de automatización de las cadenas de fabricación. La primera patente de un dispositivo robótico fue solicitada en marzo de 1954 por el inventor británico C.W. Kenward. Dicha patente fue emitida en el Reino Unido en 1957. Sin embargo, fue George C. Devol, ingeniero norteamericano, inventor y autor de varias patentes, el que estableció la bases del robot industrial moderno.

En 1954 Devol diseña y patenta el *"primer robot industrial programable"* al que denomina *"dispositivo de transferencia programada de artículos"*. Devol afirmó que el robot industrial *"ayudaría al trabajador de las fábricas del mismo modo en que las máquinas de ofimática habían ayudado al oficinista"*.

En 1958 Devol pone esta idea en conocimiento de Joseph F. Engelberger, y comienzan a trabajar en la utilización industrial de sus máquinas, fundando la *Consolidated Controls Corporation*, que más tarde se convierte en *Unimation* (Universal Automation). En 1961 instalaron su primera máquina *Unimate* (Máquina de Transferencia Universal) que fue usado en la producción lineal de la planta de *General Motors* en la localidad de Trenton (Nueva Jersey), en una aplicación de carga y descarga de piezas en una máquina de fundición por inyección ya que fabricaba puertas, manivelas de ventanas, palancas de cambios, luces y otros componentes para el interior de los coches. Este robot estaba controlado por interruptores de fin de carrera y levas, utilizando motores hidráulicos y obedeciendo los comandos almacenados en un tambor magnético.

Figura 1-1. Primer robot Unimate.

Otras grandes empresas como la *AMF* (American Machine and Foundry), emprendieron la construcción de máquinas similares como el primer robot cilíndrico llamado *Versatran*.

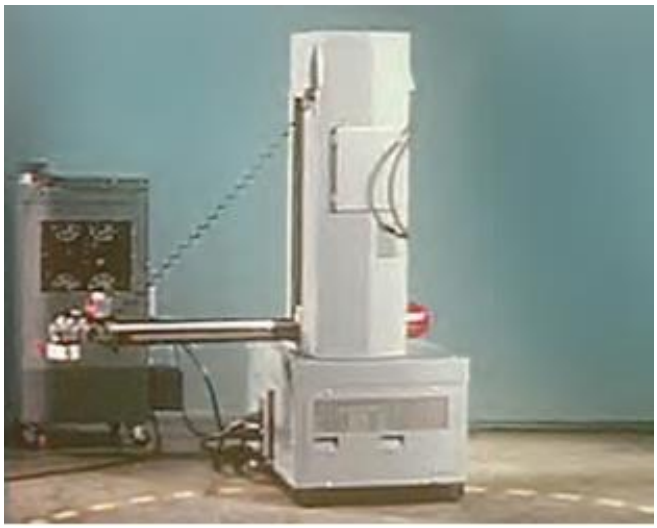


Figura 1-2. Robot Versatran, primer robot cilíndrico AMF.

Por motivos comerciales, se empezó a denominar robots a estas máquinas destinadas a transferir piezas de manera *"versátil"* y *"universal"*, a pesar de que su aspecto no era el humanoide de los robots de la literatura o del cine. Este cambio de nombre, favoreció la difusión y aceptación de los robots por parte de la industria, proyectando una imagen de modernismo y avance tecnológico. La *Ford Motor Company*, uno de los primeros usuarios que jugó un importante papel en el desarrollo de la robotica, se resistió a adoptar este nombre debido a sus connotaciones literarias, denominándolos UTD (Universal Transfer Device).

En 1964 se crean departamentos de investigación en Inteligencia Artificial en el *MIT* (Massachusetts Institute of Technology), el *SRI* (Standford Research Institute) y la Universidad de Edimburgo. También se diseñan los primeros robots móviles con un cierto grado de autonomía.

Los robots *Unimation* impulsaron la productividad y permitieron la automatización de más del 90% de las soldaduras.

Figura 1-3. Robot de soldadura por puntos en la fábrica de General Motors.

El año 1970 fue importante en la historia de la robótica debido al comienzo de la primera conferencia internacional sobre robots llamada *"Simposio Nacional de Robots Industriales"* impartido en Chicago (USA). El propósito de este simposio era proporcionar a los investigadores e ingenieros de todo el mundo una oportunidad para presentar su trabajo y compartir sus ideas en los campos de la robótica.

Los 70 fue una década de grandes avances en el campo de la robótica con acontecimientos como la creación de la primera asociación de robótica del mundo: *JIRA* (Japanese Industrial Robot Association), lo que permitió un desarrollo espectacular de la robótica en Japón.

También cabe destacar en esta década la construcción del primer robot del mundo con accionamiento totalmente eléctrico, el robot IRB6 que fue adquirido por *Magnussons* en Genarp para encerar y pulir tubos de acero inoxidable de mano de la empresa sueca ASEA; posteriormente dicha empresa ya convertida en la actual *ABB*, se convertiría en una de las empresas más importantes del mundo en la fabricación de robots industriales, y Suecia uno de los países con más robots *per capita*.

Figura 1-4. Robot IRB6 adquirido por Magnussons.

También tuvo lugar en 1973 la producción del primer robot de seis ejes impulsados electromecánicamente cuyo nombre fue "*Famulus"*. *KUKA* partió de los robots *Unimate* para desarrollar sus propios robots.

Figura 1-5. Robot Famulus de KUKA.

El 1974 creó el *Instituto de Robótica de América* (RIA- Robotics Industries Association).

La empresa *Unimation* construye en el año 1978 el robot industrial hasta ahora más famoso y utilizado tanto en ambientes industriales como de investigación, el robot *PUMA* (Programmable Universal Machine for Assembly) que se utilizaba en las tareas industriales de montaje fundamentalmente y se caracterizaba por su tamaño similar al operador humano ya que ambos ocupaban el mismo espacio de trabajo.

Figura 1-6. Robot PUMA de Unimation.

En los 80 se funda la *IFR* (Federación Internacional de Robótica) y la *AER* (Asociación Española de Robótica). Otro hito tecnológico importante en el diseño de robots es el accionamiento directo, desarrollado en la *Universidad Carnegie Mellon* en 1981. Los motores se acoplan directamente a las articulaciones sin necesidad de reductores, lo que permite movimientos más rápidos y precisos.

La configuración de los primeros robots respondía a las denominadas configuraciones esféricas y antropomórficas, de uso especialmente válido para la manipulación. En 1982 el profesor Hiroshi Makino de la *Universidad Yamanashi de Japón*, desarrolló y construyó el primer robot *SCARA* (Selective Compliance

Assembly Robot Arm); se trata de un robot de 3-4 ejes y una configuración orientada al ensamblado de piezas.

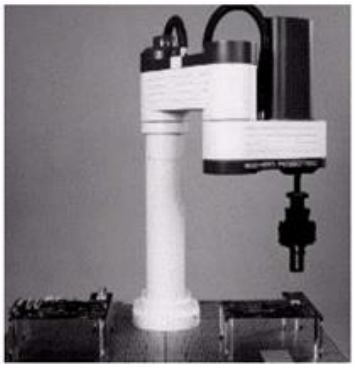


Figura 1-7. Robot SCARA.

En el periodo desde 1980 hasta 2013, el trabajo y la investigación en robótica, realizada por empresas y por universidades, se orienta al desarrollo de sensores para el robot, cada vez más potentes, que permitan lograr un comportamiento *"inteligente"* del robot, para su integración en el entorno, y para poder tomar decisiones en tiempo real. También se construyen robots que utilizan accionamiento eléctrico y tienen capacidades de carga muy elevadas, pudiendo manipular piezas de hasta 500 kg.

El tiempo transcurrido desde el comienzo de la robótica ha contemplado un intenso desarrollo, sobre todo en la robótica industrial, de tal forma que los robots, que llegaron a ser considerados el paradigma de la automatización industrial, se han convertido en nuestros días en un elemento más, aunque importante, de dicha automatización. Los robots sugieren modernidad y avance científico, reflejando en épocas pasadas, presentes, y seguramente futuras, muchas de las expectativas del progreso tecnológico.

En la actualidad los robots industriales son un elemento imprescindible para sustentar en gran medida la disposición de bienes de consumo de cuya fabricación masiva son piezas clave.

1.2. Definición de robot industrial

Hoy en día no existe una única definición formal de lo que es un robot industrial. La existencia de múltiples definiciones, muchas de ellas diferenciadas en ligeros matices, genera cierta confusión. La Federación Internacional de Robótica (IFR), distingue al robot industrial de manipulación de otros robots definiéndolo de la siguiente manera:

"Robot industrial de manipulación es una máquina de manipulación automática, reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento"

Aunque, sin duda, la definición más acertada y que deja patente la dificultad de definir qué es un robot la proporciona **Joseph Engelberg, padre de la robótica industrial:**

"Puede que no sea capaz de definirlo, pero sé cuando veo uno".

1.3. Aplicaciones de un robot industrial

La aplicación de los robots para manejo de objetos ofrece gran potencial para librar a la mano de obra humana de trabajos monótonos, cansados o peligrosos. Incluye la transferencia de piezas entre sistemas de bandas transportadoras o líneas de procesos en los que las partes pueden ser pesadas, estar calientes, tener propiedades abrasivas o incluso ser radiactivas.

Los robots tienen una amplia gama de aplicaciones en la industria. Actualmente la mayoría de las aplicaciones se dividen fundamentalmente en tres categorías:

1. ***Aplicaciones de manipulación de objetos y de carga y descarga de máquina:*** Aquí la función del robot es desplazar materiales o piezas de un lugar a otro dentro de la célula de trabajo. Dentro de la actividad de manipulación de materiales está incluida la carga y/o descarga de una máquina de producción.
2. ***Aplicaciones de procesos:*** Incluye la soldadura por puntos, la soldadura por arco, la pintura por pulverización y otras operaciones donde el robot manipula una herramienta dentro de la célula de trabajo.
3. **Montaje e inspección:** El montaje robótico es un campo en que la industria tiene gran interés debido a su potencial económico. Los robots de inspección utilizan sensores para calibrar y medir características de calidad del producto fabricado.

Las ventajas de los robots frente al operador humano son las siguientes:

- Pueden ser más fuertes, lo que permite levantar pesos considerables y aplicar mayores fuerzas.
- No presentan problemas psicológicos por monotonía ni cansancio y pueden trabajar fácilmente las veinticuatro horas del día. No necesitan descansos.
- Son consistentes. Una vez que se han instruido para realizar un trabajo pueden repetirlo con un alto grado de precisión. El desempeño humano tiende a deteriorarse con el paso del tiempo.
- Son casi completamente inmunes a su ambiente. Pueden trabajar en entornos extremadamente fríos o calientes, o en áreas donde existe el peligro de gases tóxicos o radiación. Manipulan objetos con temperaturas muy elevadas. Son capaces de trabajar en la oscuridad.
- Los robots producen niveles de calidad superiores y trabajan a mayor velocidad que los aprendices y que los oficiales de trabajo. Se pueden programar para que realicen la tarea de operarios cualificados. Esto dará origen a fábricas con automatización flexible donde los robots realizarán el trabajo de seres humanos.

En cambio no todo son ventajas en la utilización de robots en la industria. El hombre supera al robot cuando la tarea a realizar es compleja, es decir, cuando es decisiva la visión tridimensional, el uso repetido del tacto, la intervención de dos manos simultáneamente, o es necesario un gran número de decisiones en un tiempo muy corto.

1.4. Especificaciones de un robot industrial

Cuando se decide la adquisición de un robot industrial es necesario realizar un cuidadoso estudio de las especificaciones técnicas que acompañan al sistema (manipulador y unidad de control). Estas características dan una idea del comportamiento dinámico del brazo (posicionamiento, velocidad, aceleración) y de las capacidades de programación y control (memoria, lenguaje, comunicaciones con elementos externos).

En los catálogos de los fabricantes es frecuente ver una descripción más o menos homogénea, citando parámetros y características parecidos. Sin embargo, existe el inconveniente de que la interpretación de determinados conceptos puede ser diferente según el fabricante de que se trate. Por ello a continuación se explicará los conceptos más habituales que definen las características mecánicas y funcionales de un robot, utilizando las definiciones más comúnmente aceptadas en el mundo de la robótica.

2 MANIPULADOR IRB120 DE ABB

EL modelo IRB 120 es el robot industrial multiusos de 6 grados de libertad más pequeño de ABB tan solo pesa 25 kg y puede soportar una carga de 3 kg (4 kg en posición vertical de la muñeca) con un alcance de 580 mm. Es una elección fiable y rentable para generar una gran capacidad de producción a cambio de una baja inversión.

Es ideal para aplicaciones de manipulación de materiales y de montaje, gracias a que proporciona una solución ágil, compacta y ligera, con un control superior y la precisión del recorrido. Se puede montar en cualquier ángulo sin ninguna restricción

Está diseñado específicamente para industrias de fabricación que utilizan una automatización flexible basada en robots. El robot tiene una estructura abierta especialmente adaptada para un uso flexible y presenta unas grandes posibilidades de comunicación con sistemas externos. Principalmente es usado para el manejo de pequeños materiales en la industria farmacéutica y electrónica.

Es compacto y está alimentado por una unidad de control de movimientos de altas prestaciones. Cuenta con una estructura ligera, de aluminio; además dispone de unos potentes motores compactos que aseguran que el robot está habilitado con una aceleración rápida, y pueden ofrecer una precisión y agilidad en cualquier aplicación.

Dicho robot está equipado con el controlador IRC5 y el software de control de robots RobotWare. RobotWare y BaseWare OS controlan todos los aspectos del sistema de robot, como el control de los movimientos, el desarrollo y la ejecución de programas de aplicación, la comunicación... Para disponer de una funcionalidad mayor, es posible equipar al robot con software opcional para compatibilidad con determinadas aplicaciones, como la aplicación de adhesivo o la soldadura al arco, funciones de comunicación o comunicaciones de red, además de funciones avanzadas como el procesamiento multitarea, el control de sensores, etc.

2.1 Especificaciones

Los datos del robot industrial IRB 120 que proporciona el fabricante ABB son los siguientes

Tabla 2–1 Especificaciones Robot IRB 120

ESPECIFICACIONES			
Versiones de robot	Alcance	Carga Útil de la muñeca	Carga del brazo
IRB120-3/0.6	580mm	3Kg (4Kg muñeca en posición vertical)	0.3kg

Tabla 2–2 Caracterisiticas Robot IRB 120

CARACTERISTICAS	
Suministro de señal integrada	10 señales en la muñeca
Suministro de aire integrado	4 de aire en la muñeca (5 bar)
Posición Repetibilidad	0.01mm
Montaje del robot	Cualquier ángulo
Grado de protección	IP30
controladores	IRC5 Compact / IRC5 Single cabinet

Tabla 2–3 Movimientos Robot IRB 120

MOVIMIENTOS		
Ejes de movimiento	Rango de trabajo	Velocidad máxima IRB 120
Eje 1. Rotación	+165º a -165º	250 º/s
Eje 2. Brazo	+110º a -110º	250 º/s
Eje 3. Brazo	+70º a -110	250 º/s
Eje 4. Muñeca	+160º a -160º	320 º/s
Eje 5. Balanceo	+120º a -120º	320 º/s
Eje 6. Giro	+400º a -400º	420 º/s

Tabla 2–4 conexiones eléctricas Robot IRB 120

CONEXIONES ELECTRICAS	
Tensión de alimentación	200-600V, 50/60 Hz
Potencia nominal	3.0 kVA
Potencia consumida	0.25 kW

Tabla 2–5 Caracteristicas físicas del Robot IRB 120

CARACTERISITICAS FISICAS	
Dimensión base robot	180x180mm
Dimensión cuerpo robot	700mm
peso	25 Kg

Tabla 2–6 Tiempo de actuación del Robot IRB 120

TIEMPO DE ACTUACION	
1kg *Picking cycle*	IR120
25x300x25mm	0.58s
25x300x25mm con 180 ° eje 6 reorientación	0.92 s
Tiempo de aceleración 0-1 m/s	0.07 s

Al tratarse de un modelo de gran rapidez hay que tener en cuenta los sobrecalentamientos producidos en aplicaciones que requieren movimientos fuertes y frecuentes

Tabla 2–7 Temperatura ambiente para el Robot IRB 120

AMBIENTE	
Temperatura ambiente para el robot manipulador	
Durante la operación	Desde +5ºC hasta +45ºC
Transporte y almacenamiento relativo	Desde +5ºC hasta +45ºC
Para periodos cortos	Hasta +70ºC
Humedad relativa	Max 95%
Nivel de ruido	Max 70 db(A)
Seguridad	Parada de emergencia y de seguridad

2. canal de supervisión circuitos de seguridad
3. dispositivo habilitación de posición

El modelo IRB 120 puede soportar una carga de 3 kg (4 kg en posición vertical de la muñeca) con un alcance de 580 mm.

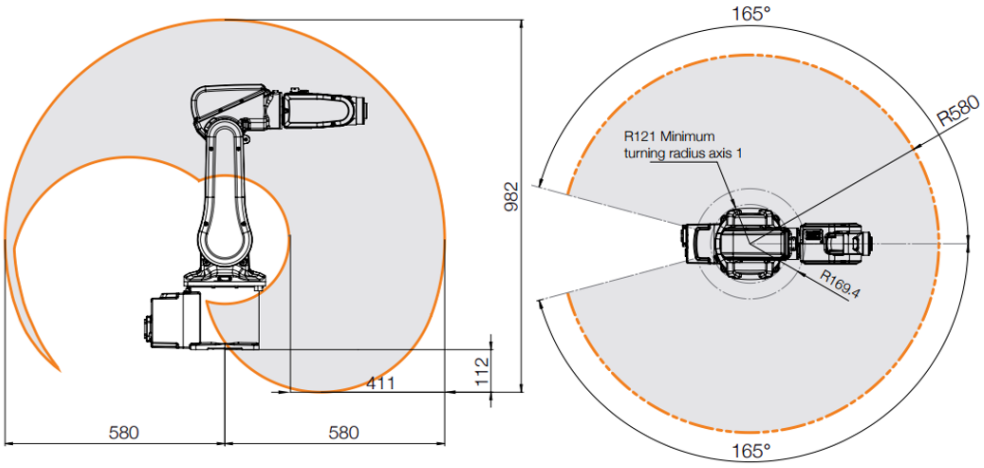


Figura 2-1.Area de trabajo del robot IRB 120 de ABB.

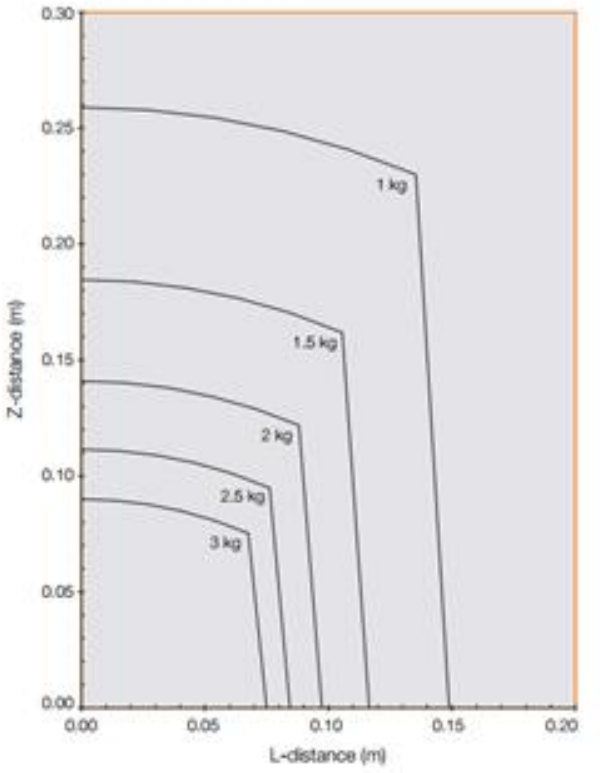


Figura 2-2. Carga útil de la muñeca del robot IRB 120 de ABB.

3 CELULA DE TRABAJO

3.1 Entorno de trabajo del manipulador

El laboratorio de robótica de la universidad de Sevilla cuenta con un robot ABB IRB-120, es un robot manipulador que permite manipular materiales son establecer un contacto directo. Cuenta con seis grados de libertad permitiendo que cualquier posición pueda ser alcanzada con cualquier orientación.

Una célula típica de robot ABB está compuesto por el siguiente hadware.

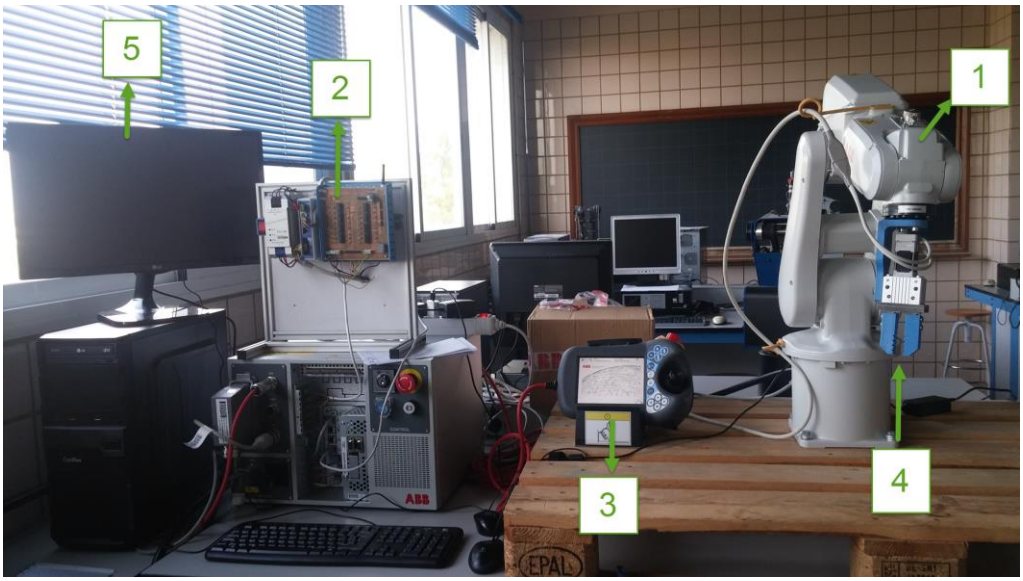


Figura 3-1. Célula de trabajo real. Manipulador IRB 120 de ABB

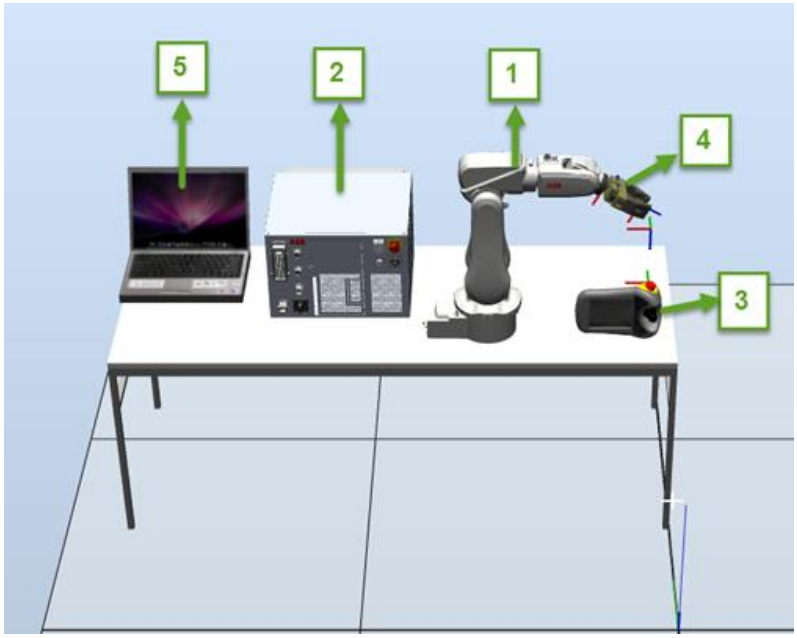


Figura 3-2. Célula de trabajo en RobotStudio. Manipulador IRB 120 de ABB.

1. ***Manipulador de robot:*** robot industrial de ABB modelo IRB 120
2. ***FlexController:*** el armario de controlador de los robots IRC5. Se compone de un módulo de control y un módulo de accionamiento para cada manipulador de robot del sistema.
3. ***FlexPendant:*** La unidad de programación conectada al módulo de control.
4. ***Herramienta:*** un dispositivo montado normalmente en el manipulador de robot para que éste pueda realizar determinadas tareas.
5. ***Ordenador:*** Dispositivo que cuenta con un software específico para la simulación y el manejo del manipulador con el cual se puede trabajar tanto online como offline.

3.2 Puesta en marcha del manipulador real

La puesta en marcha del manipulador se llevará a cabo mediante el *FlexController IRC5. El selector deberá estar ubicado en la zona de ON,* tal como se ve en la ampliación de la imagen.

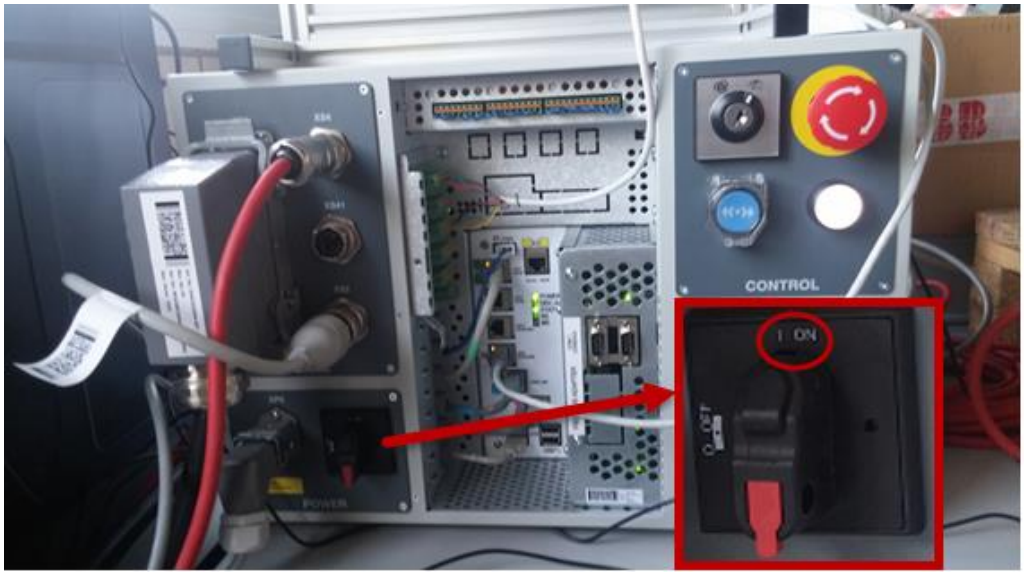


Figura 3-3.Puesta en marcha del controlador IRC5.

Una vez esté en funcionamiento el manipulador el Flexpendant comenzará a cargarse para poder trabajar con él.

Figura 3-4.Carga del Flexpendant.

Para poder ver el menú se deberá desplegar el icono de listado.

Figura 3-5. Menú desplegable.

Una vez desplegado a parecerán las diferentes opciones de menú que dispondremos para poder trabajar con el robot.

Figura 3-6. Opciones del Menú desplegable.

Las opciones que más se emplearán serán:

- Movimiento : a través del cual podremos elegir como realizar movimientos de manera manual a través del joystick

Figura 3-7. Opción movimientos.

- Editor de programa: a través del cual se podrán programar una serie de rutinas e instrucciones para manipular el robot de manera automática

Figura 3-8. Opción Editor de programa.

Es importante tener en cuenta que para poder accionar los motores será necesario mantener pulsado el botón de "hombre muerto". Dicho botón dispone de tres posiciones pero solo en la segunda posición será cuando los motores estén habilitados, el resto de posiciones son de seguridad por tanto los bloquearan. Además el flexpendant dispone de una seta de emergencia accesible en todo momento (el controlador IRC5 también dispone de una seta de emergencia)

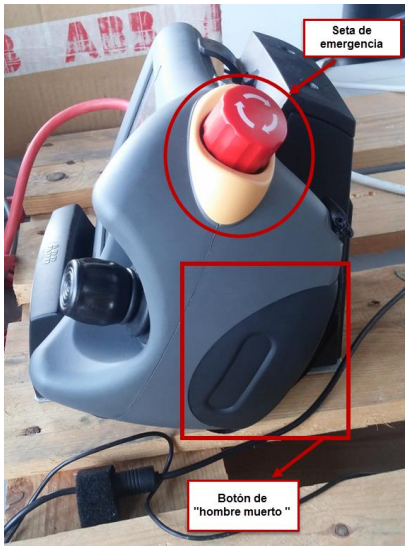


Figura 3-9. Botón de "hombre muerto".

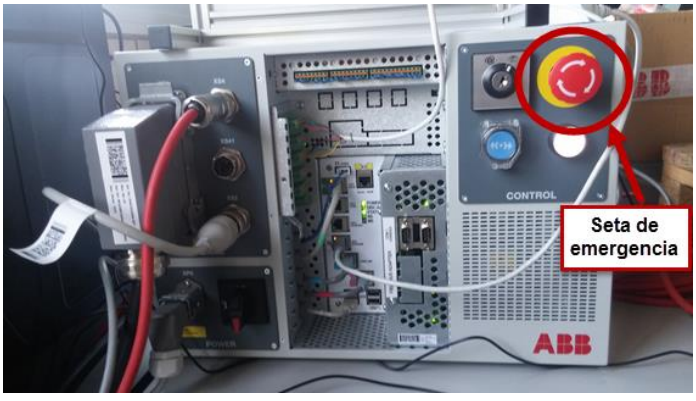


Figura 3-10. Seta de emergencia.

Se sabrá si los motores están habilitados cuando se ilumine el led ubicado en la zona de control del flexcontroller

Figura 3-11. Motores apagados.

Figura 3-12. Motores habilitados.

También se podrá sincronizar el manipulador a través del software de ABB " RobotStudio" donde se desarrollará el programa RAPID a través del editor para su posterior simulación antes de volcar los datos al manipulador real (el manejo de dicho programa se detallará en los siguientes capítulos).

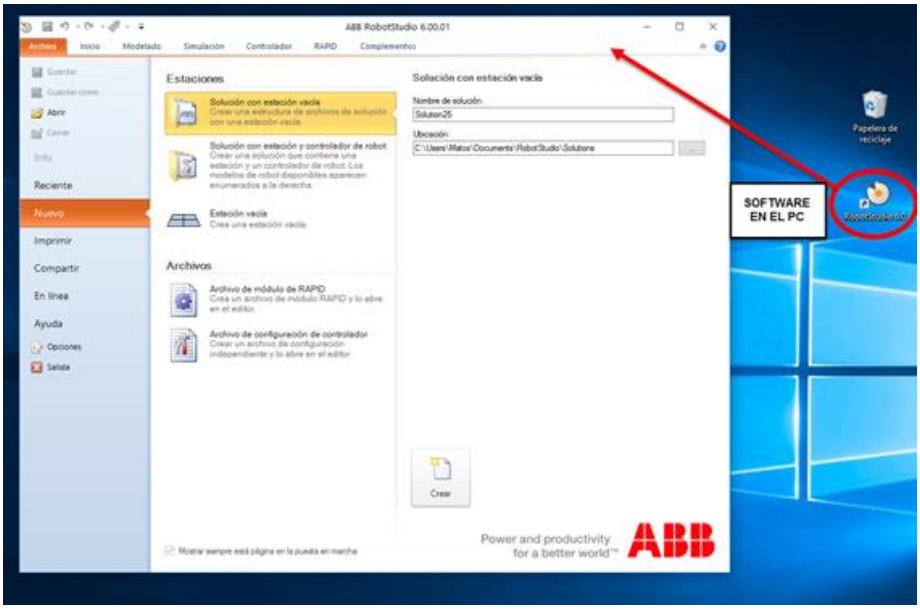


Figura 3-13. Programa RobotStudio.

4 PROGRAMACIÓN RAPID

La programación del robot es la forma que tiene el usuario de indicar la secuencia de operaciones que debe realizar el robot para llevar a cabo la aplicación. Es en su facilidad de reprogramación, donde radica la ventaja de la utilización de robots como dispositivos de fabricación flexibles.

A diferencia de los lenguajes de programación con ordenadores, en robótica cada fabricante de robot ha desarrollado su propio lenguaje de control de sus modelos, potenciando las funcionalidades de su sistema, ya que los lenguajes clásicos de la informática no disponen de las instrucciones y comandos específicos que necesitan los robots, para aproximarse a su configuración y a los trabajos que han de realizar. Ni siquiera existe normalización en cuanto a los procedimientos de programación de robots, por lo que suele ser difícil establecer una comparación entre ellos.

Generalmente hablando, hay tres maneras de comunicarse con un robot que son:

- Reconocimiento de voz.
- Enseñanza y repetición.
- Leguajes de programación de alto nivel.

Es imprescindible que los lenguajes para los robots sean fácilmente ampliables, por lo que se les debe dotar de una estructura modular, con inclusión de subrutinas definidas por el mismo usuario.

En este proyecto solo se empleará el lenguaje RAPID (Robotics Application Programming Dialogue),ya que solo usaremos un robot de ABB en nuestras aplicaciones. Se trata de un lenguaje de programación textual de alto nivel, desarrollado por ABB en 1994 para sustituir al ARLA. Sus características más destacadas son el uso de procedimientos, rutinas parametrizables, funciones y rutinas TRAP que se ejecutan cuando se activa una interrupción, estructura modular del programa y posibilidad de declarar rutinas y datos como locales o globales. Además de ello cuenta con el uso de estructuras predefinidas para especificar la configuración del robot y las características de la herramienta.

ABB presenta dos tipos de modos de operación: manual y automático. En el modo manual pueden realizarse todas las operaciones de edición, ejecución y supervisión; permite trabajar a nivel de usuario (no se necesitan conocimientos de la sintaxis de programación ya que se confeccionan los programas guiados por menú) o programador experto. La ejecución del modo automático , asegura que la velocidad va a ser el 100% de la programada; cuando el usuario cambia de modo automático a manual la velocidad disminuye un porcentaje determinado respecto a la programada; es una medida de seguridad , ya que se supone que el usuario va a trabajar sobre el programa.

4.1 Conceptos de programación de manipuladores

A la hora de programar los movimientos del robot hay que saber diferenciar entre los objetivos y las trayectorias

4.1.1 Objetivo

Es una coordenada que debe ser alcanzada por el robot. Un objetivo contiene la siguiente informacion

- Posicion : La posición del objetivo, definida en un sistema de coordenadas del objeto de trabajo.
- Orientacion: La orientación del objetivo, respecto de la orientación del objeto de trabajo. Cuando el robot alcanza el objetivo, alinea la orientación del TCP con la orientación del objetivo.

- Configuración: Valores de configuración que especifican la forma en que el robot debe alcanzar el objetivo.

4.1.2 Trayectoria

Una secuencia de instrucciones de movimiento. Las trayectorias se utilizan para hacer que el robot se mueva a lo largo de una secuencia de objetivos

4.1.3 Multimove

Se denomina multimove a la ejecución de varios manipuladores de robot con el mismo módulo de control.

4.1.4 Posición de destino

La posición de destino de una instrucción de movimiento se define en forma de coordenadas de un sistema de coordenadas. Si no se especifica ningún sistema de coordenadas, la posición se indica de forma relativa al sistema de coordenadas de la base del robot (también conocido como base de coordenadas de la base).

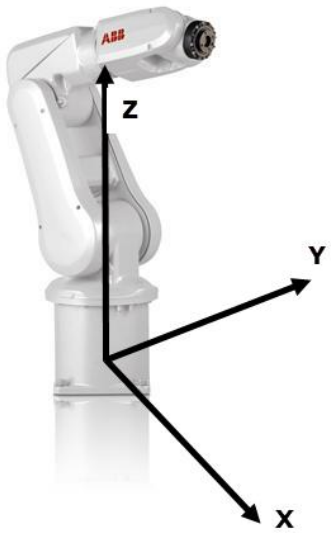


Figura 4-1. Base de coodenadas de la base.

Es posible definir y utilizar otro sistema de coordenadas con las instrucciones de movimiento. El sistema de coordenadas que la instrucción de movimiento debe utilizar se especifica con el argumento opcional [\Wobj]

4.1.5 Sistema de coordenadas

El sistema de coordenadas se utiliza para definir posiciones y orientaciones. Al programar un robot, es posible aprovechar distintos sistemas de coordenadas para posicionar más fácilmente los objetos uno respecto de otro.

4.1.6 Objeto de trabajo

El objeto de trabajo (wobj) representa normalmente a la pieza de trabajo física.

4.1.7 Sistema de coordenadas del objeto de trabajo

A la hora de programar un robot, todos los objetivos (posiciones) dependende la base de coordenadas de un objeto de trabajo. Si no se especifica ningún objeto de trabajo, los objetivos dependen del objeto predeterminado Wobj0, que siempre coincide con la base de coordenadas de la base del robot.

Las utilidades del sistema de coordenadas del objeto de trabajo son:

- Cuando la posición de la pieza respecto del robot en la estación real no coincide exactamente con su posición en la estación offline
- Cuando se realizan movimientos coordinados

El sistema de coordenadas del objeto de trabajo se compone de dos sistemas de coordenadas: la base de coordenadas del usuario (se utilizan para crear puntos de referencia a elección del usuario para simplificar la programación) y la base de coordenadas del objeto (se utilizan para definir los objetivos para proceder a la configuración de los ejes. Para distinguir entre las distintas configuraciones, todos los objetivos tienen un valor de configuración que especifica en qué cuadrante debe situarse cada eje)

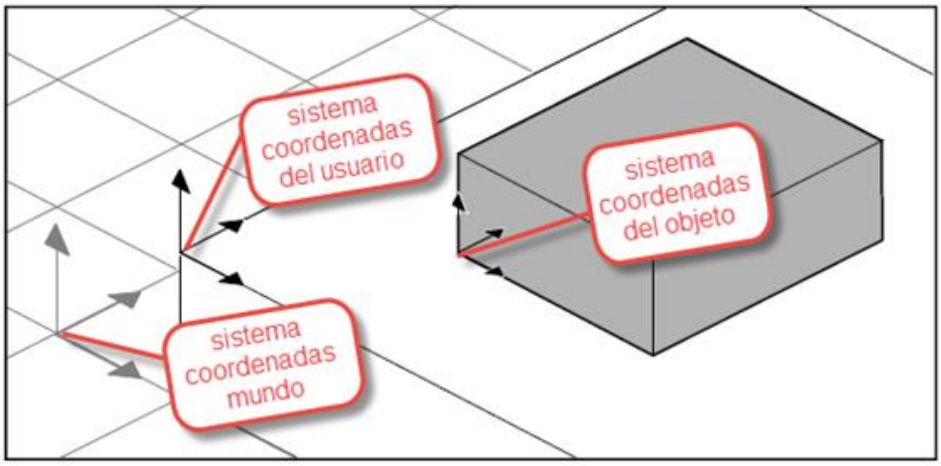


Figura 4-2. Composición del sistema de coodenadas del objeto de trabajo.

4.1.8 Sistema de coordenadas del punto central de la herramienta

El sistema de coordenadas del punto central de la herramienta, denominado también TCP, es el punto situado en el centro de la herramienta. Es posible definir distintos TCP para un mismo robot. Todos los robots tienen un TCP predefinido en el punto de montaje de la herramienta en el robot, identificado como tool0.

4.1.9 Sistema de coordenadas de RobotStudio

Sistema de coordenadas mundo de RobotStudio (RS-WCS): El sistema de coordenadas mundo de RobotStudio representa a la totalidad de la estación o célula de robot. Se encuentra en la parte superior de la jerarquía de la que dependen todos los demás sistemas de coordenadas (si se utiliza RobotStudio).

Base de coordenadas de la base (BF): El sistema de coordenadas de la base se denomina base de coordenadas de la base (BF). Cada robot de la estación, tanto en RobotStudio como en el mundo real, tiene un sistema de coordenadas de la base que siempre está situado en la base del robot.

Base de coordenadas de la tarea (TF): La base de coordenadas de la tarea representa el origen del sistema de coordenadas mundo del controlador de robot en RobotStudio.

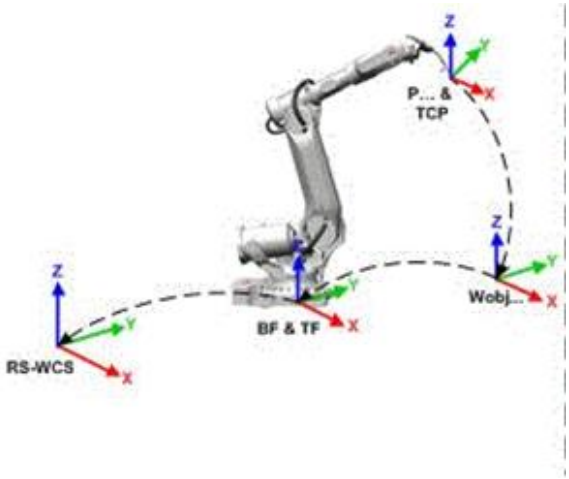


Figura 4-3. Base de coordenadas TF ubicada en la misma posiciónque la base de coordenadas BF.

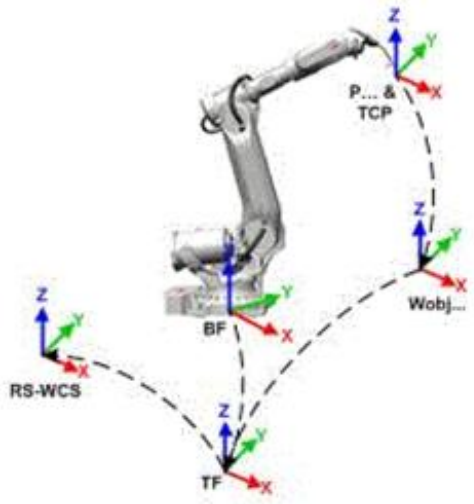


Figura 4-4. Base de coordenadas TF traslada a otra posición.

4.1.10 Estaciones con multiples sistemas de robot

En el caso de un sistema de robot individual, la base de coordenadas de la tarea de RobotStudio se corresponde con el sistema de coordenadas mundo del controlador. Si hay varios controladores presentes en la estación, la base de coordenadas de la tarea permite el trabajo de los robots en sistemas de coordenadas diferentes.

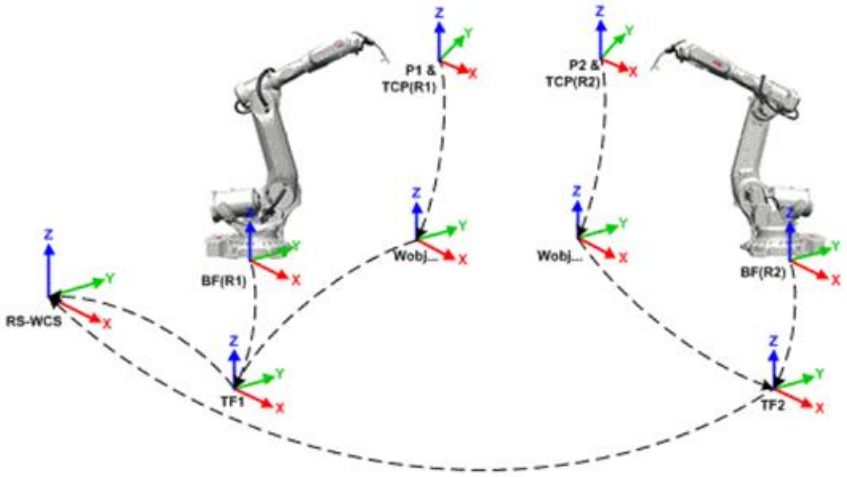


Figura 4-5. Sistema de coordenadas con multiples robots.

4.1.11 Sistema multimove coordinated

Las funciones de MultiMove le permiten crear y optimizar programas para sistemas MultiMove en los que un robot o posicionador sostiene la pieza de trabajo mientras otros robots trabajan en ella. Es importante que los robots funcionen en el mismo sistema de coordenadas

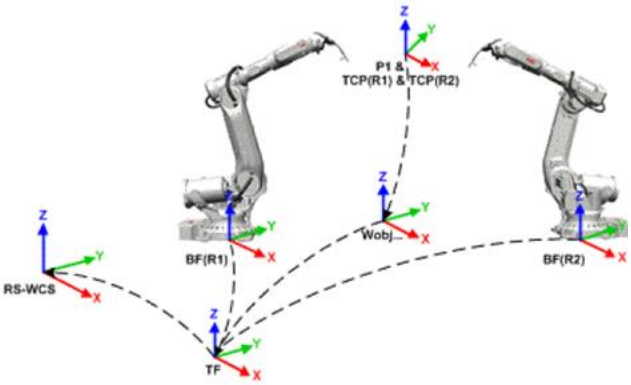


Figura 4-6. Sistema multimove coordinated con 2 manipuladores.

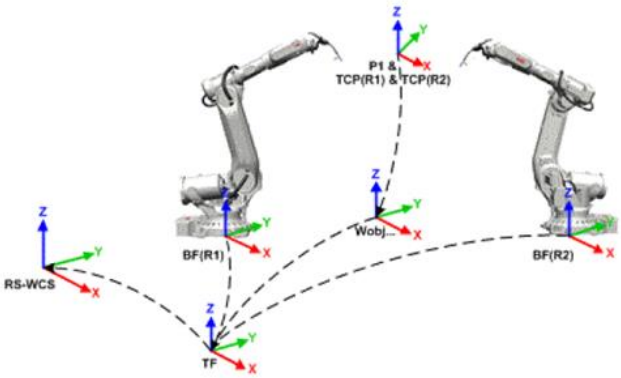


Figura 4-7. Sistema multimove coordinated con 3 manipuladores.

4.1.12 Sistema multimove Independet

En el caso de un sistema de robot con la opción de RobotWare MultiMove Independent, los robots funcionan de forma simultánea e independiente, mientras son controlados por un único controlador. Aunque sólo hay un sistema de coordenadas mundo en el controlador de robot, frecuentemente los robots trabajan con sistemas de coordenadas de tarea separadas y pueden posicionarse de forma independiente entre sí.

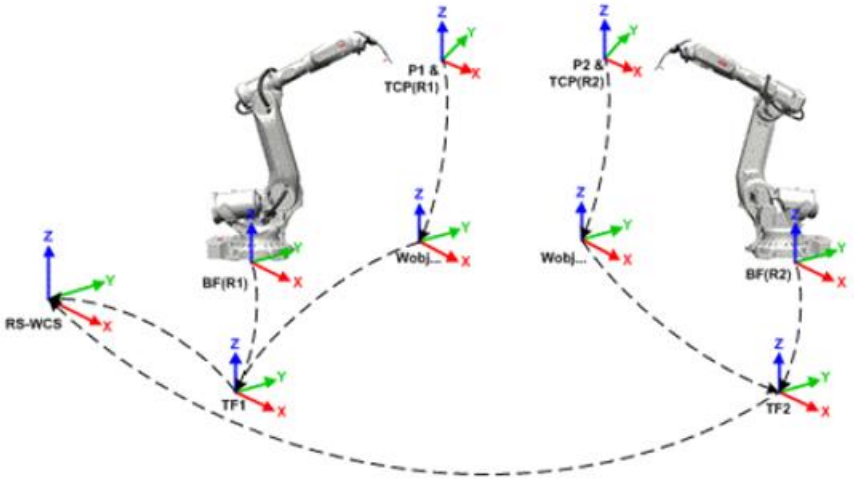


Figura 4-8. Sistema molvimove independet con 2 manipuladores.

4.1.13 Modos de programación

Antes de comenzar a programar se debe tener claro los diferentes modos de programación de los que se dispone.

- **Programación en línea**: programación con conexión a un controlador real. Esta expresión también implica el uso del robot para crear posiciones y movimiento.
- **Programación fuera de línea:** programación sin una conexión al robot ni al controlador real
- **Programación real fuera de línea:** Se refiere al concepto creado por ABB Robotics en el sentido de conectar un entorno de simulación a un controlador virtual. Esto no sólo permite crear programas, sino también realizar las pruebas y la optimización de los programas fuera de línea.

Tabla 4–1 Conceptos de programación

Concepto	Explicación
Declaración de datos	Se utilizan para crear instancias de variables o tipos de datos, como num o tooldata.
Instrucción	Los comandos de código en sí que hacen que ocurra algo, por ejemplo el cambio de un dato a un valor determinado o un movimiento del robot. Las instrucciones sólo pueden ser creadas dentro de una rutina.
Instrucciones de movimiento	Crean los movimientos del robot. Se componen de una referencia a un objetivo, especificado en una declaración de datos, junto con parámetros que establecen el comportamiento de los movimientos y procesos. Si se usan objetivos en línea, la posición se declara en las instrucciones de movimiento.
Instrucción de acción	Puede utilizarse para definir y cambiar parámetros. Se insertan antes, después o entre objetivos de instrucción en las trayectorias
Rutina	Normalmente, un conjunto de declaraciones de datos, seguido de un conjunto de instrucciones utilizadas para implementar una tarea. Las rutinas pueden dividirse en tres categorías: procedimientos, funciones y rutinas TRAP.
Procedimiento	Un conjunto de instrucciones que no devuelve ningún valor
Función	Un conjunto de instrucciones que devuelve un valor
Rutina TRAP	Un conjunto de instrucciones que es disparado por una interrupción.
Módulo	Un conjunto de declaraciones de datos, seguido de un conjunto de rutinas. Los módulos pueden ser guardados, cargados y copiados

	en forma de archivos. Los módulos se dividen entre módulos de programa y módulos de sistema.
Módulo de programa (.mod)	Pueden ser cargados y descargados durante la ejecución.
Módulo de sistema (.sys)	Se utilizan principalmente para datos y rutinas específicos del sistema, por ejemplo un módulo de sistema de ArcWare que es común para todos los robots de soldadura al arco.
Archivos de programa (.pgf)	En IRC5, un programa de RAPID es una colección de archivos de módulo (.mod) y el archivo de programa (.pgf.) que hace referencia a todos los archivos de módulo. Al cargar un archivo de programa, todos los módulos de programa antiguos se reemplazan por aquéllos a los que se hace referencia en el archivo .pgf. Los módulos de sistema no se ven afectados por la carga de programas.

4.2 Estructura de un programa en Rapid

Un programa robot está formado por una serie de instrucciones que describen el trabajo del mismo: instrucciones de movimiento, de entradas y salidas, se comunicación, etc. Las instrucciones contienen argumentos asociados (valor numérico, referencia de un dato, expresión de una función o valor de cadena) que definen lo que va a producir la ejecución de la instrucción.

Todo programa en RAPID dispone de un encabezado de archivo, se trata de un archivo de programa que empieza siempre con el siguiente encabezado:

```
%%%
VERSION: 1                    %Versión del sistema
LANGUAGE: ENGLISH             %Idioma
%%%
```

Figura 4-9. Encabezado archivo del programa rapid.

Una aplicación consta de un programa y una serie de módulos de sistema. A su vez el programa puede estar dividido en varios módulos

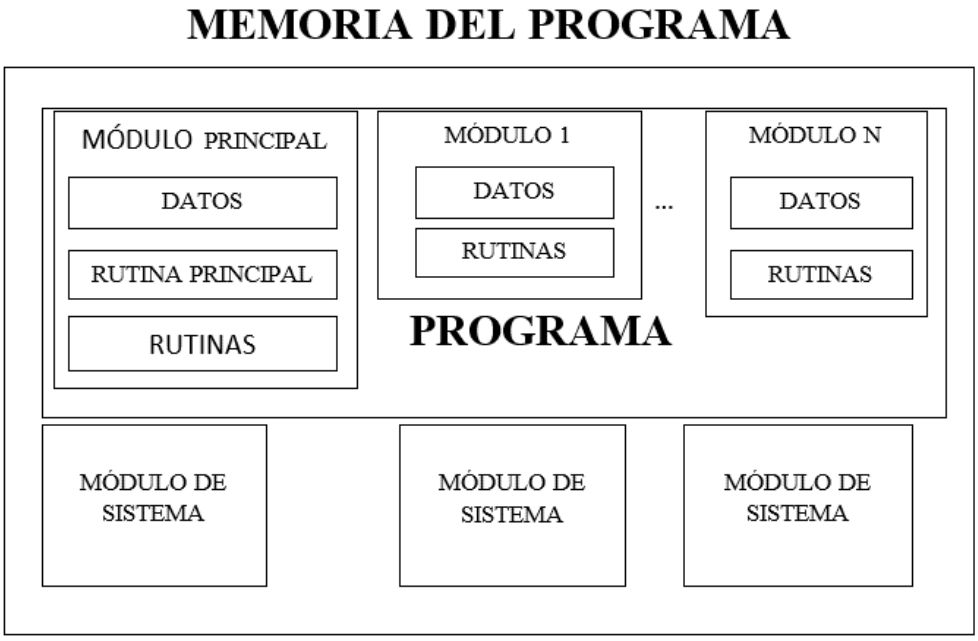


Figura 4-10. Memoria del programa rapid.

El programa es una secuencia de instrucciones que controlan el robot y en general constan de tres partes:

- Rutina principal (main): Rutina donde se inicia la ejecución del programa
- Conjunto de sub-rutinas: sirven para dividir el programa en partes más pequeñas a fin de obtener un programa modular.
- Datos del programa: definen posiciones, valores numéricos, sistemas de coordenadas…

Cada módulo puede ser cargado o salvado independientemente.

Para el controlador del robot no supone ninguna diferencia que el programa este escrito en distintos módulos. El motivo de utilizar distintos módulos de programa es sólo facilitar a los programadores la comprensión del programa y la reutilización del código.

Al declarar un módulo hay que especificar un nombre y unos atributos (los atributos solo se pueden añadir OFF-LINE). Los tipos de atributos más comunes son los siguientes:

- SYSMODULE: se trata de un módulo de sistema.
- NOSTEPIN: no permite la entrada en el módulo durante la ejecución paso a paso.
- VIEWONLY: el módulo no puede ser modificado.
- READONLY: el módulo no puede ser modificado pero sus atributos puedes ser eliminados.
- NOVIEW: no puede ser visualizado, sólo ejecutado. Un programa que contiene un módulo como este no puede ser salvado. Se suele usar para los módulos de sistema.

La sintaxis utilizada para asignar los atributos al módulo es la siguiente:

```
MODULE modulo1 (SYSMODULE, NOVIEW)

! Declaraciones
!Instrucciones
ENDMODULE
```

Figura 4-11. Sintaxis para atributos de módulos.

Los módulos de sistema sirven para la definición de los datos y rutinas específicas del sistema. Estos módulos no serán incluidos cuando se salva un programa, lo que significa que una modificación realizada en un módulo de sistema afectará a todos los programas. Dichos módulos residen siempre en memoria (se cargan durante el arranque en frío del sistema) y pueden ser utilizados siempre.

Una rutina típica tiene la siguiente estructura:

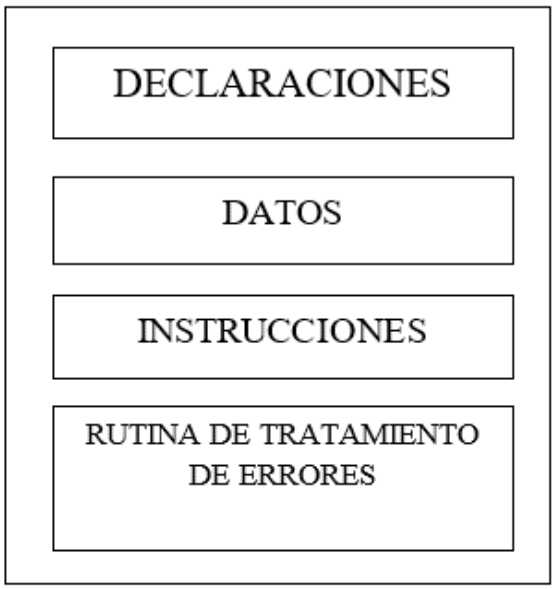


Figura 4-12. Estructura de una rutina.

Existen tres tipos de rutinas a la hora de realizar un programa:

- **Procedimientos:** rutinas que no devuelven ningún valor.

```
PROC nombre_proc        (…argumentos…)
```

```
…
ENDPROC
```

El final de un procedimiento está indicado por ENDPROC, BACKWARD o ERROR, pero también se puede forzar con la instrucción RETURN.

- **Funciones:** devuelven un valor de un tipo determinado

```
FUNC tipo_devuelto nombre_func (…argumentos…)
…
RETURN (…);
…
ENDFUNC
```

La evaluación de una función debe terminar con una instrucción RETURN.

- **Rutinas de tratamiento de interrupciones:** para el tratamiento de las interrupciones. Nunca pueden ser llamadas explícitamente desde el programa.

```
TRAP nombre_rut_trat_int
…
RETURN
…
ENDTRAP
```

El final de una rutina de interrupción está indicado por ENDTRAP o ERROR, pero también se puede forzar con la instrucción RETURN.

Los parámetros de una rutina son los argumentos que se deben suministrar a dicha rutina cuando se la llama. Tipos de parámetros:

- **Normales:** se utilizan como entradas a la rutina y son procesados como variables de la rutina. El cambio de esta variable en la rutina no cambia el valor del argumento.
- **Inout:** el argumento correspondiente debe ser una variable o un entero persistente y su valor puede ser cambiado por la rutina.
- **Var:** el argumento correspondiente debe ser una variable que puede ser cambiada por la rutina.

- **Pers:** el argumento correspondiente debe ser un persistente entero que puede ser cambiado por la rutina.

Un parámetro puede ser opcional. En ese caso se antepone un \ en la declaración. Si hay varios parámetros opcionales mutuamente excluyentes entre sí, se deben separar por |.

```
PROC rutina1 (num par_in, INOUT num par_inout, VAR num
              par_var,
 PERS num par_pers, num par_obligat, \num par_opcional,
                                     \num par_opc_1 | num par_opc_2)
```

Un tipo especial switch sólo puede ser asignado a parámetros opcionales y proporciona una forma de utilizar argumentos especificados por los nombres y no por los valores

```
PROC rutina2 (\switch on | \switch off)
…
IF present  (off) THEN
…
ENDPROC
```

Las matrices también pueden utilizarse como argumento. La dimensión de la matriz será determinada por el parámetro.

```
PROC rutina3 (VAR num matriz1{*,*})
```

En algunos casos es necesario interrumpir la ejecución secuencial de las instrucciones del programa. El flujo del programa se puede controlar mediante

- **Llamadas a rutinas**
  - *ProcCall*: llamada (salto) a otra rutina.
  - *CallByVar*: llamada a un procedimiento con nombre específico.
  - *RETURN*: regreso a la rutina original.

- **Instrucciones de evaluación de condición**
  - *CompactIF*: ejecuta una instrucción sólo si se cumple una condición.
  - *IF*: puede usarse cuando un conjunto de sentencias sólo debe ejecutarse si se cumple una condición determinada. Si la condición lógica se cumple, el código de programa situado entre las palabras claves *THEN* y *ENDIF* se ejecuta. Si la condición no se cumple, ese código no se ejecuta y la ejecución continúa después de *ENDIF*.
  - *ELSE*: una sentencia *IF* también puede contener código de programa para su ejecución si la condición no se cumple. Si la condición lógica de la sentencia *IF* se cumple, el código

de programa situado entre las palabras claves *THEN* y *ELSE* se ejecuta. Si la condición no se cumple, el código situado entre las palabras clave *ELSE* y *ENDIF* se ejecuta.
- *ELSEIF*: en ocasiones existen más de dos secuencias de programa alternativas. En este caso puede usarse ELSEIF para crear distintas alternativas.
- *WHILE*: repite una parte del programa hasta que se cumpla una condición.
- *TEST*: ejecuta diferentes partes del programa según el valor de una expresión.

- **Instrucciones de repetición de una secuencia**
  - *FOR*: repite una secuencia de código del programa un cierto número de veces.

- **Salto a una etiqueta dentro de una rutina**
  - *GOTO*: salta a una etiqueta.

- **Parada de la ejecución del programa**
  - *STOP*: detiene la ejecución del programa.
  - *EXIT*: para la ejecución de un programa cuando el rearranque de un programa no está permitido.
  - *BREAK*: para la ejecución de un programa temporalmente para el diagnostico y la solución de averías.

4.3 Fundamentos del lenguaje Rapid

El lenguaje RAPID es un lenguaje sin formatos, lo que significa que los espacios, tabuladores, caracteres fin de línea o fin de página pueden utilizarse en cualquier parte excepto en identificadores, palabras reservadas, valores numéricos o comodines. La regla general a tener en cuenta a la hora de realizar un programa es que cada instrucción termina con un punto y coma ";" excepto para algunas instrucciones especiales que en lugar de terminar en punto y coma, existen palabras clave especiales para indicar dónde terminan. Como por ejemplo:

Tabla 4–2 Palabras clave de instrucciones

Palabra clave de instrucción	Palabra clave final
IF	*ENDIF*
FOR	*ENDFOR*
WHILE	*ENDWHILE*
PROC	*ENDPROC*

Los identificadores sirven para nombrar módulos, rutinas, datos y etiquetas. El primer carácter de un identificador debe ser una letra, el resto de caracteres pueden ser letras, números o el carácter de subrayado "_", sin sobrepasar la longitud máxima de 16 caracteres. Los identificadores no diferencian si es mayúscula o minúscula pero hay que tener cuidado con una serie de palabras que no pueden ser utilizadas como identificadores, son aquellas palabras reservadas del lenguaje RAPID (AND, IF, MODULE, LOCAL, etc.) o los nombres predefinidos para tipos de datos del sistema, instrucciones o funciones.

Los comentarios se incluyen para facilitar la comprensión del programa y no afectan en modo alguno al

funcionamiento del mismo. Un comentario siempre empieza con el símbolo "!" y acaba con el carácter fin de línea.

Los comodines se utilizan para representar de forma temporal ciertas partes del programa que todavía no han sido definidas. Los comodines más empleados en programación son los siguientes.

Tabla 4–3 Comodines más utilizados en la programación Rapid

Comodín	Representa
<DDN>	Una declaración de datos
<RDN>	Una declaración de rutina
<PAR>	Un parámetro alternativo opcional
<ALT>	Un parámetro opcional
<DIM>	Una definición de la dimensión de una matriz
<SMT>	Una instrucción
<VAR>	Una referencia (variable, persistente o parámetro) a un dato
<EIT>	La cláusula ELSE de una instrucción IF
<CSE>	La cláusula CASE de una instrucción TEST
<EXP>	Una expresión
<ARG>	Un argumento de llamada de procedimiento
<ID>	Un identificador

RAPID cuenta con muchos tipos de datos diferente, pero todos se basan en los tres tipos de datos generales. Con ellos se podrá realizar diferentes operaciones y combinarlos para formar tipos de datos más complejos. Los tipos de datos generales son:

- **Variables:** contienen valores de datos. Al detener el programa y volverlo a poner en marcha, la variable conserva su valor, pero si se mueve el puntero de programa a Main el valor del dato de la variable se pierde. Se le puede asignar un nuevo valor durante la ejecución del programa. Los tres tipos más conocidos son :
    - Valor numérico [num]: podrá ser entero o coma flotante. Deberá estar dentro de los límites especificados por el estándar ANSI de coma flotante simple precisión.
    - Valor lógico [bool]: se trata de una variable booleana, podrá expresarse como TRUE (verdadero) o FALSE (falso).

- Valor de cadena [string]: es una secuencia de caracteres (ISO 8859-1) y de caracteres de control. Se pueden incluir códigos de caracteres precedidos por “ \ ” y caracteres no imprimibles. Una cadena puede tener como máximo 80 caracteres y va siempre entre comillas “ ” .

Los datos de variable con tipo de valor podrán ser inicializados en la declaración. La declaración de una variable es la forma de definir un nombre de una variable y determinar el tipo de dato que debe tener. Las variables se declaran con la palabra clave VAR, siguiendo la sintaxis:

```
VAR tipo_dato nombre;
```

A su vez la asignación de un valor a una variable se hace con la instrucción := como se puede observar en el siguiente ejemplo:

```
VAR num length;

length:=10;
```

- **Persistentes:** es una variable que tiene la particularidad de que su valor de inicialización se actualiza a medida que va cambiando. Cuando se guarda un programa, el valor de inicialización de cualquier declaración de persistente refleja el último valor que ha tomado el dato persistente. Las variables persistentes se declaran con la palabra clave PERS. En el momento de la declaración es necesario indicar un valor inicial.

```
PERS tipo_dato nombre:= valor_inic
```

Así como se muestra en el próximo ejemplo.

```
PERS num nbr := 1;
```

Los datos persistentes sólo pueden declararse en el nivel de módulo (fuera de las rutinas) y deben tener obligatoriamente un valor inicial. Si un dato persistente es actualizado, se actualizará automáticamente su valor de inicialización.

- **Constantes:** el valor se asigna siempre en el momento de la declaración y posteriormente no es posible cambiar el valor en ningún caso. La constante se declara con la palabra clave CONST seguida del tipo de dato, el identificador y la asignación de un valor.

```
CONST tipo_dato nombre := valor;
```

Cuyo ejemplo sería:

```
CONST num gravedad := 9.81;
```

El alcance de una rutina o módulo hace referencia a desde dónde se puede acceder a dicha rutina. Desde este punto de vista hay dos tipos:

- **Locales:** sólo se puede acceder desde el módulo que la contiene. Se deben declarar añadiendo al principio la palabra reservada LOCAL.

```
[LOCAL] VAR tipo_dato nombre;
```

- **Globales:** puede ser llamada desde cualquier módulo. Las rutinas son globales por defecto.

Atendiendo a su descomposición, existen dos grandes grupos de tipos de datos:

- **Atómicos:** datos cuya definición no está basada en otro tipo y que no puede ser dividido en partes ni componentes.
- **Registro:** tipo de datos formado por componentes ordenados y nombrados.

Atendiendo a su valor podemos considerar otros dos grupos diferentes.

- **Con valor:** representan de alguna forma un valor. Pueden ser utilizados en operaciones que tratan con valores: inicialización, asignación, comparación, etc.
- **Sin valor:** no representan un valor.

Existen tres grupos de operadores empleados para enlazar diferente tipos de datos.

- **Operadores numéricos:** estos operadores operan con el tipo de datos num y devuelven el tipo de datos num.

Tabla 4–4 Operadores numéricos

Operador	Descripción	Ejemplo
+	Suma	reg1 := reg2+reg3
-	Resta	reg1 := reg2-reg3 reg1 := -reg2
*	Multiplicación	reg1 := reg2*reg3
/	División	reg1 := reg2/reg3

- **Operadores relacionales**: estos operadores devuelven un tipo de dato bool.

Tabla 4–5 Operadores relacionales

Operador	Descripción	Ejemplo
=	Igual a	Flag1:= reg1= reg2 Flag1 es TRUE si reg1 es igual a reg2
<	Menor que	Flag1:= reg1< reg2 Flag1 es TRUE si reg1 es menor que reg2
>	Mayor que	Flag1:= reg1> reg2 Flag1 es TRUE si reg1 es mayor que reg2
<=	Menor o igual que	Flag1:= reg1<= reg2 Flag1 es TRUE si reg1 es menor o igual que reg2
>=	Mayor o igual que	Flag1:= reg1>= reg2 Flag1 es TRUE si reg1 es mayor o igual que reg2
< >	Distinto de	Flag1:= reg1< > reg2 Flag1 es TRUE si reg1 es distinto de reg2

- **Operador de cadena:** para enlazar varios datos string.

Tabla 4–6 Operadores de cadena

Operador	Descripción	Ejemplo
+	Concatenación de cadenas	VAR string firstname:= “John”; VAR string lastname:= “Smith”; VAR string fullname; fullname := firstname+” “+lastname; La variable fullname contendrá la cadena “John

Smith”

A continuación se mostrará la lista de los diferentes tipos de datos que más adelante detallaremos.

Tabla 4–7 Tipos de datos más empleados en la programación Rapid

Bool	Valores lógicos
Clock	Medida del tiempo
Confdata	Datos de configuración del robot
Dionum	Valores digitales (0,1)
Errnum	Numero de error
Extjoint	Posición de los ejes externos
Intnum	Identificación de una interrupción
Iodev	Canales y archivos de comunicación serie
Jointtarget	Datos de posición de los ejes
Loaddata	Datos de carga
Mecunit	Unidad mecánica
Motsetdata	Datos de movimiento
Num	Valores numéricos (registros)
Orient	Orientación
O_lointtarget	Dato de posición original de un eje
O_robtarget	Dato de posición original
Pos	Posiciones (x,y,z) sin orientación
Pose	Transformación de coordenadas
Progdisp	Desplazamiento de programa

Robjoint	Posición de los ejes del robot
Robtarget	Datos de posición (completos)
Signalai	Señales de entrada analógicas
Signalao	Señales de salida analógicas
Signaldi	Señales de entrada digitales
Signaldo	Señales de salida digitales
Signalgi	Grupo de señales de entrada digitales
Signalgo	Grupo de señales de salida digitales
Speeddata	Datos de velocidad
String	Cadena de caracteres
Symnum	Número simbólico
Tooldata	Datos de herramienta
Triggdata	Eventos de posicionamiento (disparo)
Tunetype	Tipo de ajuste servo
Wobjdata	Datos del objeto de trabajo
Zonedata	Datos de zona de ajuste

Seguidamente se describirán más detalladamente los tipos de datos más usuales de la programación de estos robots.

- Pos: se usa para almacenar la posición del robot (x,y,z).Sus componentes son :

Tabla 4–8 Componentes tipo de datos Pos

X	Coordenada X	Num
Y	Coordenada Y	Num

Z	Coordenada Z	Num

VAR pos posic1:= [600,500,400];

- Orient: se utiliza para almacenar la orientación de un sistema de coordenadas. Sus componentes son las siguientes:

Tabla 4–9 Componentes tipo Orient

q_1	Cuaternio 1	Num
q_2	Cuaternio 2	Num
q_3	Cuaternio 3	Num
q_4	Cuaternio 4	Num

La orientación de un sistema de coordenadas respecto a uno de referencia se puede describir mediante una matriz de rotación. Esta matriz tendrá por columnas los vectores normalizados que representan las direcciones de los ejes X, Y, Z respecto al sistema de coordenadas de referencia.

$$x = (x_1, x_2, x_3) \tag{4–1}$$

$$y = (y_1, y_2, y_3) \tag{4–2}$$

$$z = (z_1, z_2, z_3) \tag{4–3}$$

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \tag{4–4}$$

Mediante los cuaternios podemos representar la orientación con sólo cuatro valores. Dichos cuaternios se calculan en función de los componentes de la matriz de rotación de la siguiente forma:

$$q_1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2} \tag{4–5}$$

$$q_2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2} \tag{4–6}$$

$$q_3 = \frac{\sqrt{y_2 - x_1 - z_3 + 1}}{2} \tag{4–7}$$

$$q_4 = \frac{\sqrt{z_3 - x_1 - y_2 + 1}}{2} \tag{4–8}$$

Siempre ha de cumplirse que

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \qquad (4–9)$$

VAR orient orient_1:= [1,0,0,0];

- **Pose:** se usa para definir un sistema de coordenadas respecto a otro. Sus componentes son:

Tabla 4–10 Componentes tipo Pose

Trans	Desplazamiento (x,y,z) del sistema de coordenadas	pos
Rot	Rotación del sistema de coordenadas	Orient

VAR pose pose_1:= [[50,0,40] , [1,0,0,0]];

- **Confdata:** sirve para definir las configuraciones de los ejes del robot. A veces el robot es capaz de alcanzar la misma posición y prientacion de la herramienta mediante varias configuraciones de ejes diferentes. Para evitar estas ambigüedades se utilizará este tipo de dato, cuyas componentes son:

Tabla 4–11 Componentes tipo Confdata

cf_1	Indica el cuadrante en el que se encuentra el eje 1 (entre -4 y +3)	num
cf_4	Indica el cuadrante en el que se encuentra el eje 4 (entre -4 y +3)	num
cf_6	Indica el cuadrante en el que se encuentra el eje 6 (entre -4 y +3)	num
cf_x	No se utiliza de momento	num

VAR confdata config1:= [1,-1,0,0];

Las configuraciones de los ejes del robot se designan con una serie de cuatro números enteros que especifican en qué cuadrante de una revolución completa se encuentran los ejes significativos.

Los cuadrantes están numerados de cero en adelante para una rotación positiva (en el sentido contrario a las agujas del reloj) y de -1 en adelante para la rotación negativa (en el sentido de las agujas del reloj).

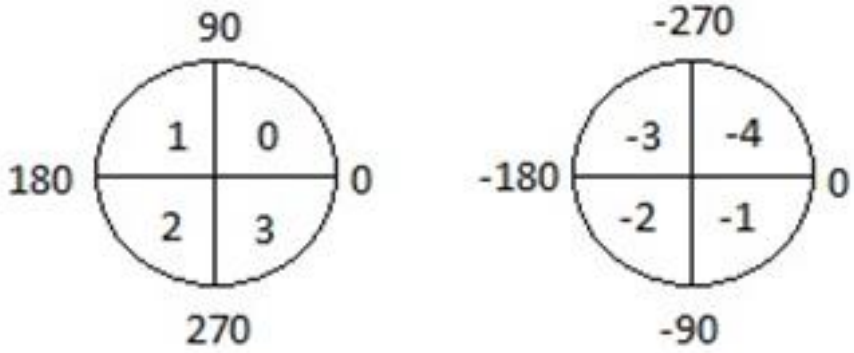


Figura 4-13. Cuadrante configuración ejes del robot.

En el caso de los ejes lineales, el entero especifica el rango (en metros) en el que se encuentra el eje desde la posición neutral.

A la hora de ejecutar un programa de robot, es posible decidir si se desea monitorizar los valores de las configuraciones mediante las instrucciones ConfL y ConfJ. Si la monitorización de configuraciones está desactivada, los valores de configuración almacenados en los objetivos no se tienen en cuenta y el robot utilizará la configuración más cercana a su configuración actual para alcanzar el objetivo. Si se activa, sólo utilizará la configuración especificada para alcanzar los objetivos.

- **Loaddata:** sirve para definir las cargas instaladas en la brida de montaje del robot. Las cargas definidas sirven para determinar un modelo de dinámica del robot, de forma que los movimientos puedan ser controlados de la mejor manera posible. Si la carga declarada es mayor que la real, el robot no será utilizado a su capacidad máxima, la precisión será incorrecta y habrá riesgo de vibraciones. Si la carga declarada es menor que la instalada, la precisión será incorrecta y habrá también riesgo de vibraciones. Normalmente, sólo se definirá el momento de inercia cuando la distancia entre la brida de montaje y el centro de gravedad sea menor que la dimensión de la carga. La carga útil debe ser definida sólo como persistente y no dentro de una rutina. Los componentes del tipo de dato loaddata son:

Tabla 4–12 Componentes tipo de dato Loaddata

mass	El peso de la carga (herramienta) en kg	num

cog	El centro de gravedad de la carga (x,y,z) en mm	pos
aom	La orientación de los ejes del momento de la carga en el centro de gravedad	orient
i_x	El momento de inercia de la carga alrededor del eje X en kgm^2	num
i_y	El momento de inercia de la carga alrededor del eje Y en kgm^2	num
i_z	El momento de inercia de la carga alrededor del eje Z en kgm^2	num

PERS loaddata pinza:=[5,[50,0,50],[1,0,0,0],0,0,0];

- **Speeddata:** sirve para especificar la velocidad del TCP, de la reorientación de la herramienta y de los ejes externos. Cuando se combinan diferentes tipos de movimientos hay una velocidad que limita todos los movimientos. La velocidad del resto de movimientos será reducida para que los movimientos acaben su ejecución al mismo tiempo. La velocidad también será restringida por la capacidad del robot. Hay que tener en cuenta que hay una serie de datos de velocidad ya definidos en el módulo del sistema BASE. Los componentes de este dato son los siguientes:

Tabla 4–13 Componentes tipo de dato Speeddata

v_tcp	Velocidad del TCP en mm/s	num
v_ori	Velocidad de la reorientación de la herramienta en º/s	num
v_leax	Velocidad de los ejes externos lineales mm/s	num
v_reax	Velocidad de los ejes externos rotativos en º/s	num

VAR speeddata v7:=[2000,30,200,15];

- **Zonedata:** una posición puede terminarse en un punto de paro o en un punto de paso. Un punto de paro significa que el robot debe alcanzar la posición programada, en un punto de paso, el robot

nunca alcanza la posición programada sino que comenzará a moverse hacia el siguiente punto cuando entre en la "zona" que se haya definido. El tamaño de la zona nunca podrá ser mayor que la mitad de la distancia a la posición anterior o siguiente más cercana.

Para cada posición se pueden definir dos zonas distintas, la zona de la trayectoria del TCP y la zona extendida de reorientación de la herramienta y ejes externos.

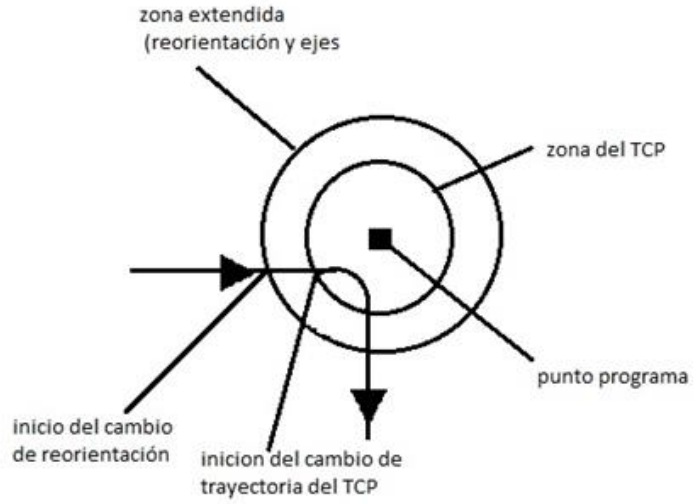


Figura 4-14.Esquema de la función Zonedata.

Los componentes de zonedata son los descritos a continuación.

Tabla 4–14 Componentes tipo de dato Zonedata

finep	TRUE: el movimiento termina en un punto de paro FALSE: el movimiento termina en un punto de paso	bool
pzone_tcp	Radio de la zona del TCP en mm	num
pzone_ori	Radio de la zona de reorientación de la herramienta en mm (debe ser mayor que pzonetcp)	num
pzone_eax	Radio de la zona de los ejes externos (debe ser mayor que pzonetcp)	num
zone_ori	Tamaño de la zona de reorientación de herramienta en grados	num
zone_leax	Tamaño de la zona de los ejes externos lineales en mm	num
zone_reax	Tamaño de la zona de los ejes externos rotativos en grados	num

Al igual que speeddata existe una serie de datos de zona ya definidos en el módulo de sistema BASE.

VAR zonedata Z25:=[FALSE,25,40,40,10,35,5];

- **Extjoint:** se utiliza para almacenar la posición de los ejes externos. Los seis ejes externos que puede controlar el robot se denominan a, b, c, d, e y f. cada uno de estos ejes lógicos podrá ser conectado a un eje físico mediante un parámetro de sistema. En el caso de que un eje lógico no esté conectado a un eje físico, se indicará 9E9 como valor de posición.
  Para los ejes rotativos, la posición será definida como la rotación en grados a partir de la posición de calibración. Para los ejes lineales, la posición será definida como la distancia en mm a partir de la posición de calibración. Los componentes de extjoint son:

Tabla 4–15 Componentes tipo de dato Exjoint

eax_a	Posición del eje externo a (en grados o mm según el tipo de eje)	num
eax_b	Posición del eje externo b (en grados o mm según el tipo de eje)	num
eax_c	Posición del eje externo c (en grados o mm según el tipo de eje)	num
eax_d	Posición del eje externo d (en grados o mm según el tipo de eje)	num
eax_e	Posición del eje externo e (en grados o mm según el tipo de eje)	num
eax_f	Posición del eje externo f (en grados o mm según el tipo de eje)	num

VAR extjoint pos_eje := [11, 25.2, 9E9, 9E9, 9E9, 9E9]

- **Robtarget:** sirve para definir la posición del robot y los ejes externos. Dado que el robot es capaz de alcanzar una misma posición de varias formas diferentes, se deberá especificar la configuración de los ejes. Sus componentes son.

Tabla 4–16 Componentes tipo de dato Robtarget

trans	Posición XYZ del TCP en mm respecto al sistema de coordenadas objeto incluyendo desplazamiento si lo hay	pos
rot	Orientación de la herramienta expresada en cuaternios	orient
robconf	Configuración de los ejes del robot (cf_1, cf_4, cf_6,cf_x)	confdata
extax	Posición de los ejes externos	extjoint

```
CONST robtarget pm1:= [[600,500,400],[1,0,0,0],[1,1,0,0][9E9, 9E9, 9E9, 9E9, 9E9, 9E9]];
```

- **Tooldata:** se utiliza para describir las características de una herramienta. Los datos de herramienta deberán ser definidos únicamente como persistentes y no deberán ser definidos dentro de una rutina.

Tabla 4–17 Componentes tipo de dato Tooldata

robhold	TRUE:el robot está sujetando la herramienta FALSE: se trata de una herramienta estacionaria	bool
tframe	Sistema de coordenadas de la herramienta: posición del TCP y orientación.(características geométricas)	pose
tload	Define la carga de la herramienta (características mecánicas de la herramienta)	loaddata

```
PERS tooldata pinza :=
```

[TRUE,[[58,26.3,0],[0.924,0,0.383,0]],[5,[23,0,75],[1,0,0,0],0,0,0]]

- **Wobjdata:** se utiliza para definir el objeto de trabajo del robot. Si se utiliza una herramienta estacionaria o ejes externos coordinados se deberá definir el objeto de trabajo, ya que la trayectoria y la velocidad se referirán al objeto de trabajo y no al TCP. Los datos de objeto deberán ser definidos únicamente como persistentes y no deberán ser definidos dentro de una rutina. Sus componentes son:

Tabla 4–18 Componentes tipo de dato wobjdara

robhold	TRUE: el robot está sujetando el objeto de trabajo FALSE: el robot está sujetando la herramienta	bool
ufprog	TRUE: sistema de coordenadas fijo del usuario FALSE: sistema de coordenadas móvil de usuario(ejes externos coordinados)	bool
ufmec	La unidad mecánica con la que los movimientos del robot están coordinados	string
uframe	Sistemas de coordenadas del usuario	pose
oframe	Sistema de coordenadas del objeto	pose

PERS wobjdata wobj1:= [FALSE,TRUE, “”,[[0,0,0],[1,0,0,0]],[[0,0,0],[1,0,0,0]]]

Las características básicas del movimiento del robot están especificadas en los argumentos de las instrucciones de movimiento. Sin embargo hay algunas que se especifican en instrucciones separadas y se aplican a todos los movimientos hasta que estas cambian. Las instrucciones de posicionamiento más usuales son las detalladas a continuación:

- ***MoveJ*:** sirve para mover rápidamente el robot de un punto a otro cuando este movimiento no tiene que realizarse necesariamente en línea recta. Todos los ejes se mueven a una velocidad constante y alcanzan su posición final a la vez. La herramienta será orientada y los ejes externos se moverán, al mismo tiempo que se mueve el TCP. En el caso en que la reorientación o los ejes externos no puedan alcanzar la velocidad programada, la velocidad de TCP será reducida.
  La sintaxis empleada para este tipo de instrucciones es:

MoveJ [\Conc] Alpunto Velocidad [\V] [\T] Zona [\Z] Herramienta [\Wobj];

MoveJ P5,v300,fine,MyTool\WObj:=wobj0;

Tabla 4–19 Componentes de la instrucción MoveJ

[\Conc]	Con esta opción, las instrucciones lógicas siguientes se ejecutarán mientras el robot está en movimiento. Acorta tiempo de ciclo	switch
Alpunto	Coordenadas y orientación del punto destino	robtarget
Velocidad	Velocidad del TCP, reorientación de la herramienta y ejes externos	speeddata
[\V]	Sirve para especificar la velocidad del TCP en mm/s directamente en la instrucción	num
[\T]	Sirve para especificar el tiempo total del movimiento en segundos	num
Zona	Tamaño de la "zona cero" del movimiento	zonedata
[\Z]	Sirve para especificar el tamaño en mm. De la "zona cero" directamente en la instrucción	num
Herramienta	Es la herramienta activa durante el movimiento. El TCP de la misma es el punto que se mueve	tooldata
[\Wobj]	Es el objeto de trabajo (sistema de coordenadas) al que se refiere la posición del robot en la instrucción. Si se omite este argumento, se refiere al sistema de coordenadas mundo	wobjdata

- ***MoveL*:** sirve para mover el TCP de la herramienta de forma lineal al punto destino. La herramienta es orientada a intervalos iguales sobre la trayectoria y los ejes externos (no coordinados) se moverán a una velocidad constante para que puedan llegar al punto de destino al mismo tiempo que los ejes del robot. En el caso en que la reorientación o los ejes externos no puedan alcanzar la velocidad programada, la velocidad de TCP será reducida. La sintaxis empleada será similar a la de *MoveJ*.

MoveL [\Conc] Alpunto Velocidad [\V] [\T] Zona [\Z] Herramienta [\Wobj];

MoveL P5,v300,fine,MyTool\WObj:=wobj0;

- ***MoveC*:** sirve para mover el TCP de la herramienta de forma circular. La herramienta es reorientada a velocidad constante desde la orientación de la posición de arranque hasta la del punto destino. La reorientación se lleva a cabo respecto a la trayectoria circular. En el caso en que la reorientación o los ejes externos no puedan alcanzar la velocidad programada, la velocidad del TCP será reducida. La sintaxis usada para este tipo de instrucciones es la siguiente:

MoveC [\Conc] PtoCirculo AlpuntoVelocidad [\V] [\T] Zona [\Z] Herramienta [\Wobj];

MoveC p1,p2,v400,fine,MyTool\WObj:=wobj0;

Tabla 4–20 Componentes de la instrucción MoveC

[\Conc]	Con esta opción, las instrucciones lógicas siguientes se ejecutarán mientras el robot está en movimiento. Acorta tiempo de ciclo	switch
PtoCirculo	Es la posición en el circulo entre el punto de arranque y de destino. No debe estar situado demasiado cerca del punto de arranque o de destino	robtarget
Alpunto	Coordenadas y orientación del punto destino	robtarget
Velocidad	Velocidad del TCP, reorientación de la herramienta y ejes externos	speeddata
[\V]	Sirve para especificar la velocidad del TCP en mm/s directamente en la instrucción	num
[\T]	Sirve para especificar el tiempo total del movimiento en segundos	num
Zona	Tamaño de la “zona cero” del movimiento	zonedata
[\Z]	Sirve para especificar el tamaño en mm. De la “zona cero” directamente en la instrucción	num

Herramienta	Es la herramienta activa durante el movimiento. El TCP de la misma es el punto que se mueve	tooldata
[\Wobj]	Es el objeto de trabajo (sistema de coordenadas) al que se refiere la posición del robot en la instrucción. Si se omite este argumento, se refiere al sistema de coordenadas mundo	wobjdata

Otras funciones de posición menos usadas son entre otras:

Tabla 4–21 Funciones de posición más utilizadas en Rapid

Offs	Añadir un offset a una expresión del robot, expresado respecto al objeto de trabajo
RelTool	Añadir un offset, expresado en el sistema de coordenadas de la herramienta
CPos	Lee la posición del robot (X,Y,Z)
CRobT	Lee la posición completa del robot
CJointT	Lee los ángulos de los ejes
ReadMotor	Lee los ángulos del motor
CTool	Lee los datos de la herramienta
CWObj	Lee los datos del objeto de trabajo
ORobT	Eliminar un desplazamiento de programa
MirPos	Realizar una copia espejo de una posición

Existe la posibilidad de definir posiciones relativas a otras posiciones ya existentes usando o bien la

instrucción *Offs* o bien *RelTool.*

Con la instrucción *Offs* hay dos maneras de definir una nueva posición respecto a un objeto de trabajo:

- Crear una nueva variable de tipo robtarget para luego definirla respecto a otra posición ya creada con anterioridad.

```
VAR robtarget pa;
Pa:= offs(pinicio,50,100,0);
```

- Se puede utilizar junto con la instrucción Move sin necesidad de definir una nueva variable

```
MoveL offs(pinicio,50,100,0),v400,z1,garra\wobj:=wobj0;
```

Con la instrucción RelTool es posible realizar movimientos respecto a otros puntos utilizando los ejes cartesianos X, Y, Z relativos a la herramienta de trabajo utilizada. Además, también se pueden realizar giros relativos a estos tres ejes.

```
MoveJ RelTool(pinicio,50,100,0,\Rz:=90),v400,z1,garra\wobj:=wobj0;
```

Se pueden dar cuatro formas distintas de comunicar a través de los canales serie:

- Mostrar mensajes que se pueden leer en el visualizador del pupitre y que pueden solicitar una respuesta del usuario.
- Leer o escribir en la memoria másica del sistema (archivos de texto)
- Transferir información binaria entre el robot y un sensor
- Transferir información binaria entre el robot y otro computador, por ejemplo un protocolo de enlace.

Si se requiere una comunicación en ambos sentidos de forma simultánea, se necesitará un tipo de transmisión binaria. Cada canal serie o archivo deberá en primer lugar ser abierto. Al hacerlo, recibirá un nombre que se utilizará posteriormente como referencia en el momento de leer o escribir. La unidad de programación esta siempre abierta y se puede utilizar en cualquier momento.

Las instrucciones más comunes para poder comunicarse con el operador a la hora de programar son las mostradas a continuación.

- **TPWrite:** escribe un texto en el visualizador del pupitre móvil.

```
TPWrite "texto" [\Num] | [\Bool] | [\Pos] | [\Orient];
```

TPWrite "Elección ilegal";

Tabla 4–22 Componentes de la instrucción TPWrite

Texto	Cadena de texto de menos de 80 caracteres	String
[\Num]	Dato numérico a añadir a la cadena	Num
[\Bool]	Dato lógico a añadir a la cadena	Bool
[\Pos]	Dato de posición del robot a añadir a la cadena	Pos
[\Orient]	Dato de orientación del robot a añadir a la cadena	orient

- **TPReadFK:** escribe una etiqueta sobre cada tecla de función y lee qué tecla ha sido pulsada.

TPReadFK contestación, "texto", "FK1", "FK2", "FK3", "FK4", "FK5" [\MaxTime] [\DIBreak] [\BreakFlag];

TPReadFK operacion, "seleccionar operación a realizar", "Cuadrado", "Circulo", "Triangulo", "", "";

Tabla 4–23 Componentes de la instrucción TPReadFK

Contestación	Variable que contendrá 1,2,3,4 o 5 según la tecla pulsada	Num
Texto	Texto informativo que aparecerá en el visualizador	String
FKx	Texto que aparece sobre las teclas de función. Máximo 7 caracteres	String
[\MaxTime]	Intervalo máximo de tiempo (en seg) que la ejecución del programa espera. Si no se pulsa ninguna tecla en ese tiempo se genera el error ERR_TP_MAXTIME	Num
[\DIBreak]	Entrada digital que podría interrumpir el diálogo. Si no se pulsa ninguna tecla antes de que esta señal	Signaldi

	pase a 1 se genera el error ERR_TP_DIBREAK	
[\BreakFlag]	Variable de salida que contiene el código de error si se utiliza MaxTIme o DIBreak. Si esta variable está presente no se produce el error	errnum

- **TPReadNum:** lee un valor numérico de la unidad de programación.

TPReadNum contestación, "texto" [\MaxTime] [\DIBreak] [\BreakFlag];

TPReadNum n_r,"Introducir numero veces a ejecutar la trayectoria ";

Tabla 4–24 Componentes de la instrucción TPreadNum

Contestación	Variable que es devuelta para el número introducido en la unidad de programación	Num
Texto	Texto informativo que se desea introducir en la unidad de programación. Máximo 80 caracteres	String
[\MaxTime]	Intervalo máximo de tiempo (en seg) que la ejecución del programa espera. Si no se introduce ningún número en ese tiempo se genera el error ERR_TP_MAXTIME	Num
[\DIBreak]	Entrada digital que podría interrumpir el diálogo. Si no se introduce ningún número antes de que esta señal pase a 1 se genera el error ERR_TP_DIBREAK	Signaldi
[\BreakFlag]	Variable de salida que contiene el código de error si se utiliza MaxTIme o DIBreak. Si esta variable está presente no se produce el error, pero si no está, el gestor de errores será ejecutado	Errnum

Otras funciones e instrucciones de dialogo con el operador comúnmente usadas son:

Tabla 4–25 Funciones de dialogo con el operador Rapid

TPErase	Borra la información del visualizador de la unidad de programación
ErrWrite	Escribe un texto en el visualizador y lo almacena en el registro de errores
Open	Abrir un canal o archivo para lectura o escritura
Write	Escribir un texto en un canal o archivo
Close	Cerrar un canal o archivo
WriteBin	Escribir en un canal serie binario
ReadNum	Sirve para leer un número a partir de un archivo de caracteres o de un canal serie
ReadStr	Sirve para leer texto a partir de un archivo de caracteres o de un canal serie

El robot está equipado con una serie de cartas de entradas y salidas. Las señales se utilizan para la comunicación con los equipos externos con los que coopera el robot. Las señales de entrada son activadas por los equipos externos y pueden usarse en el programa de RAPID para iniciar la realización de alguna operación con el robot. Las señales de salida son activadas por el programa de RAPID y actúan para comunicar a los equipos externos que deben hacer algo. Las señales son configuradas en los parámetros de sistema del sistema de robot. Es posible definir nombres personalizados para las señales. No deben declararse en el programa de RAPID.

El valor de una señal analógica o un grupo de señales digitales está especificado como un valor numérico.

Las instrucciones de E/S más usuales son:

Tabla 4–26 Instrucciones de Entradas/ Salidas

Set	Sirve para activar (poner a 1) una salida digital	Set nombre;
Reset	Sirve para desactivar (poner a 0) una salida digital	Reset nombre;
SetAO	Sirve para cambiar el valor de una salida analógica	SetAO nombre valor;

SetDO	Sirve para cambiar el valor de una salida digital con posible retraso	SetDO [\Sdelay] nombre valor;
SetGO	Sirve para cambiar el valor de un grupo de salidas digitales	SetGO nom_grupo valor;
WaitDI	Espera hasta que se active una señal de entrada digital	WaitDI nombre valor [\MaxTime] [\TimeFlag]
WaitDO	Espera hasta que se active una señal de salida digital	WaitDO nombre valor [\MaxTime] [\TimeFlag]

Las interrupciones son eventos definidos por el programa, identificados por los números de interrupción, que permite relacionar el programa con el exterior. Una interrupción se produce cuando se da una condición de interrupción. La ocurrencia de una interrupción no está directamente relacionada con una posición de una instrucción específica; provoca una suspensión de la ejecución normal del programa, y el control es transferido a una rutina de tratamiento de interrupción.

Las rutinas de tratamiento de las interrupciones proporcionan un medio para procesar las interrupciones. Es necesario conectar cada interrupción con su rutina de atención correspondiente. En el caso de que ocurra una interrupción el control es inmediatamente transferido a la rutina de tratamiento de interrupción asociada. Si ocurre una interrupción que no ha sido conectada a ninguna rutina de tratamiento, la interrupción será considerada como un error fatal, es decir, que provoca inmediatamente el paro de la ejecución del programa. Cuando se alcanza el final de la rutina hay que volver a transferir el control al programa principal y la ejecución continúa a partir del lugar en donde ocurrió la interrupción.

Cada interrupción tiene asignada una identificación que es lo que la relaciona con la rutina de tratamiento de interrupción. Esa identificación se utiliza entonces para dar la orden a una interrupción, es decir, para especificar el motivo de la interrupción, que puede ser uno de los siguientes acontecimientos:

- Una entrada o una salida está activada en 1 o en 0.
- Por estar programada para que ocurra transcurrido un tiempo.
- Que se ha alcanzado una posición específica.

Aunque el robot reconoce inmediatamente la ocurrencia de una interrupción, su respuesta (llamada a la rutina de tratamiento) sólo puede tener lugar en posiciones específicas del programa:

- Cuando se entra en la siguiente instrucción.
- En cualquier momento durante la ejecución de una instrucción de espera.
- En cualquier momento durante la ejecución de una instrucción de movimiento.

Las interrupciones se pueden habilitar e inhabilitar. Cuando se da la orden a una interrupción, ésta se encuentra automáticamente habilitada, pero podrá ser temporalmente inhabilitada. La inhabilitación se

puede producir de dos maneras:

- Todas las interrupciones pueden ser inhabilitadas: todas las interrupciones que ocurren durante este tiempo quedan almacenadas en la cola FIFO (*First Input-First Output*), y son posteriormente tratadas de forma automática, cuando vuelvan a ser habilitadas de nuevo.
- Las interrupciones pueden ser desactivadas individualmente: el sistema no tendrá en cuenta ninguna interrupción que ocurra durante este tiempo.

La cola de interrupciones puede contener un máximo de 40 interrupciones simultáneamente en espera. La gestión de la cola se realiza según el método FIFO.

Cuando se realiza una ejecución paso a paso y cuando el programa ha sido detenido, las interrupciones no serán procesadas. Las interrupciones que se producen bajo esta circunstancia, no serán tratadas.

Las instrucciones y funciones más comunes para las interrupciones son las detalladas brevemente a continuación.

Tabla 4–27 Instrucciones y funciones de las interrupciones mas usuales

Connect	Encuentra el número de identificación de la interrupción y la conecta con su rutina de tratamiento
ISignalDI	Sirve para ordenar y habilitar una interrupción a partir de una entrada digital
IsignalDO	Sirve para ordenar y habilitar una interrupción a partir de una salida digital
ITimer	Sirve para ordenar y habilitar una interrupción una vez transcurrido un tiempo
TriggIO	Define una condición de disparo para la activación de una salida en una posición determinada
TriggInt	Define una condición de disparo para la ejecución de una rutina de tratamiento de interrupciones en una posición determinada
TriggEquip	Define una condición de disparo para la activación de una salida en una posición específica con la posibilidad de incluir una compensación de tiempo por el retraso en el equipo externo
TriggC	Hacer funcionar el robot de forma circular con una condición de disparo activada
TriggJ	Hacer funcionar el robot eje a eje con una condición de disparo activada

TriggL	Hacer funcionar el robot de forma lineal con una condición de disparo activada
ISleep	Sirve para desactivar temporalmente una interrupción determinada
IDisable	Sirve para inhabilitar todas las interrupciones temporalmente
IWatch	Sirve para activar una interrupción que ha sido previamente ordenada pero que había sido desactivada mediante la instrucción ISleep
IEnable	Sirve para habilitar todas las interrupciones durante la ejecución del programa
IDelete	Sirve para anular o borrar una interrupción

5 SOFTWARE ROBOTSTUDIO

RobotStudio es un software asociado a los robots de ABB destinado al modelado, la programación tanto offline como online y la simulación de células de robot. Dicho software permite trabajar con un controlador offline, constituido con un controlador IRC5 virtual que se ejecuta localmente en su PC o un controlador real sin conexión; además también le permite trabajar en modo online con un controlador IRC5 físico real.

Robotstudio aporta herramientas que aumentan la rentabilidad del sistema de robots, pues permite realizar tareas tales como formación, programación y optimización de programas sin alterar la producción. Esto añade muchas ventajas, entre ellas:

- Reducción de riesgos
- Arranques más rápidos
- Menor tiempo para modificaciones
- Aumento de la productividad

5.1 Introducción a RobotStudio

La versión empleada en el desarrollo de este proyecto es la 5.61.01.01, ya que esta versión es de las más modernas y trabaja con robots que poseen controladores IRC5.

El procedimiento a seguir a la hora de utilizar dicho software será el siguiente:

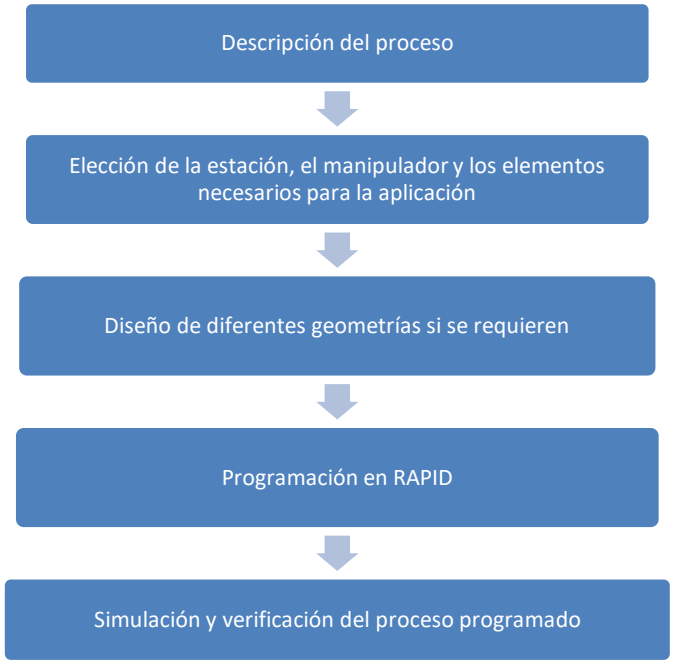


Figura 5-1.Procedimiento de utilización de RobotStudio.

El software se compone principalmente de dos archivos ejecutables.

Estación: La parte que se encarga de la creación del sistema, diseño, programación de trayectorias, movimiento manual de ejes, modelado etc. Es decir las tareas comprendidas en los menús ***Inicio*** y ***Modelado***.

Controlador virtual: La parte que se encarga del programa RAPID, edición y compilación del programa, como tarea fundamental. Es decir las tareas comprendidas en los menús ***Rapid, Controlador y Simulación***

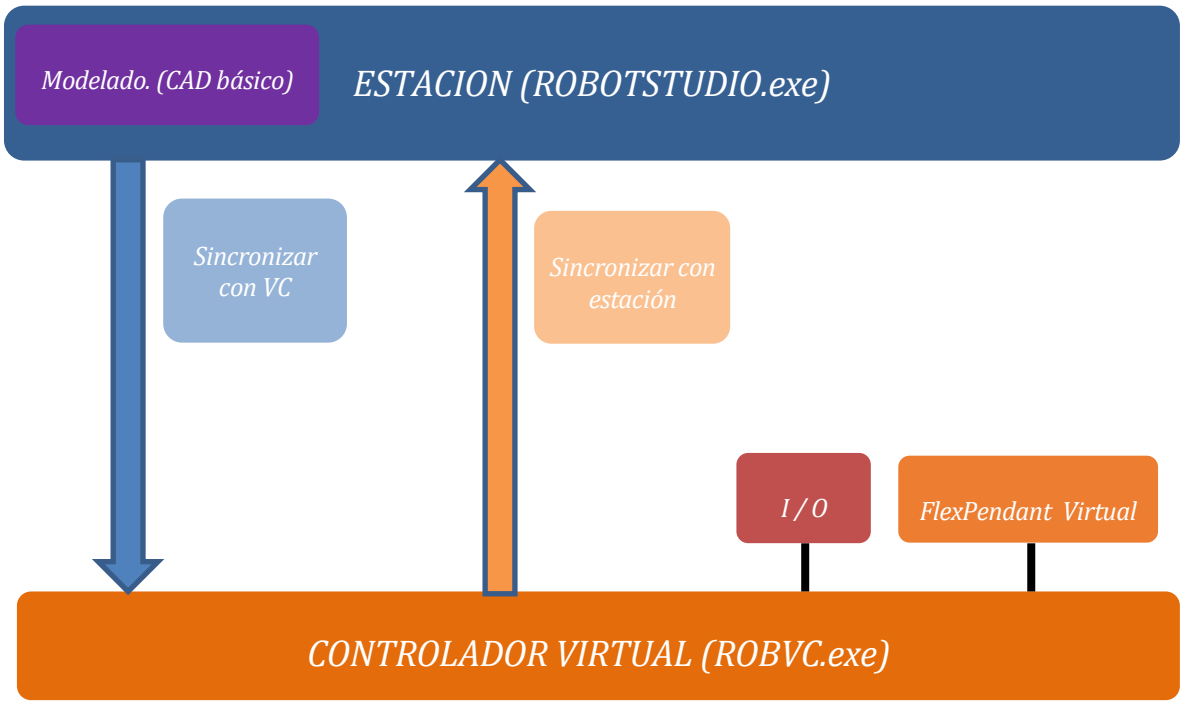


Figura 5-2.Componentes del software RobotStudio.

5.2 Interfaz gráfica

A continuación se explicarán brevemente las generalidades del entorno y las funcionalidades del mismo.

La interfaz gráfica de usuario está dividida en menús que se seleccionan activando la pestaña

Figura 5-3.Menús Interfaz gráfica.

- **Archivo:** contiene las opciones necesarias para crear una estación o un sistema de robot nuevo, guardar o abrir estaciones existentes…

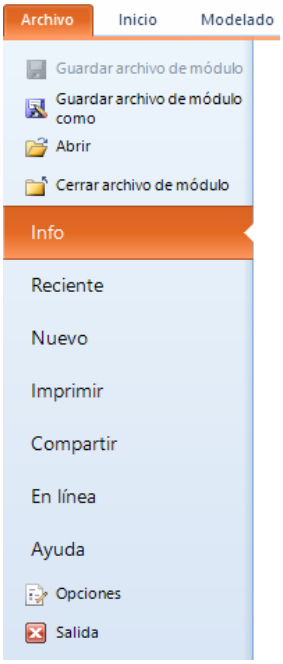


Figura 5-4.Menú Archivo.

- **Inicio:** contiene los comandos necesarios para construir estaciones, crear sistemas, programar trayectorias y colocar elementos.

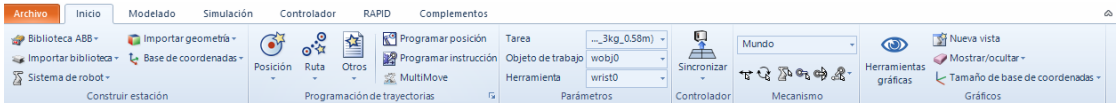


Figura 5-5.Menú Inicio.

- **Modelado:** contiene las funciones necesarias para crear y agrupar componentes, crear cuerpos, mediciones y operaciones de CAD.

Figura 5-6.Menú Modelado.

- **Simulación:** contiene los controles necesarios para crear, configurar, controlar, monitorizar, y grabar simulaciones.

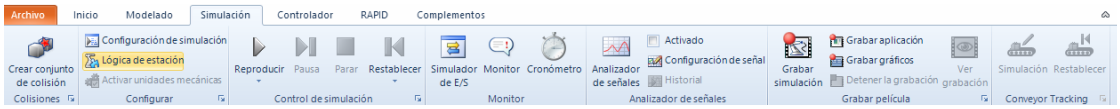


Figura 5-7.Menú Simulación.

- **Controlador:** contiene los comandos necesarios para la sincronización, configuración y tareas asignadas al controlador virtual; así como para gestionar los controladores reales.

Figura 5-8.Menú Controlador.

- **Rapid:** contiene los controladores necesarios para crear y modificar los programas necesarios del robot.

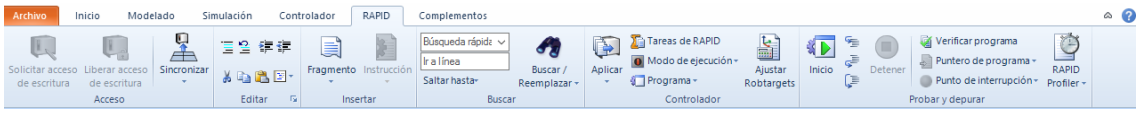


Figura 5-9.Menú Rapid.

- **Complementos:** contiene los controles de los PowerPacs y de VSTA. Además se podrán observar los valores del calor de la caja reductora y descargar componentes adicionales para las librerías del software.

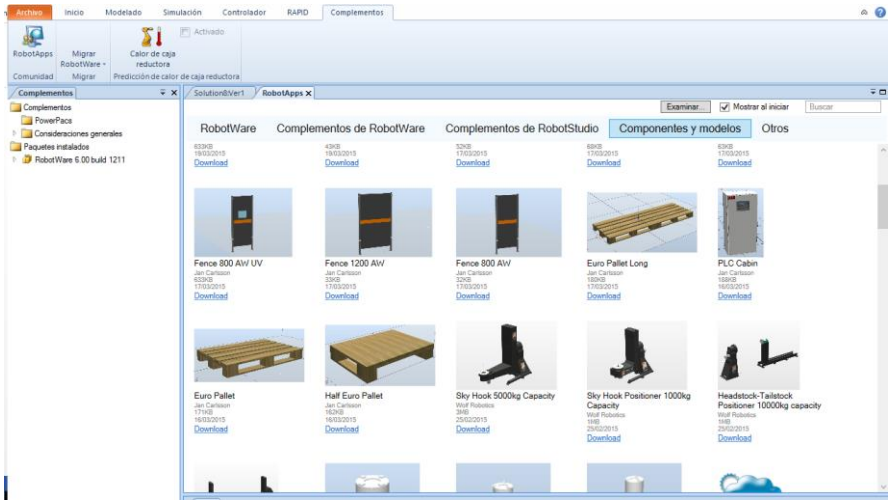


Figura 5-10.Menú Complementos.

Es necesario también conocer el entorno del sistema. Se aprecian diferentes áreas en el entorno:

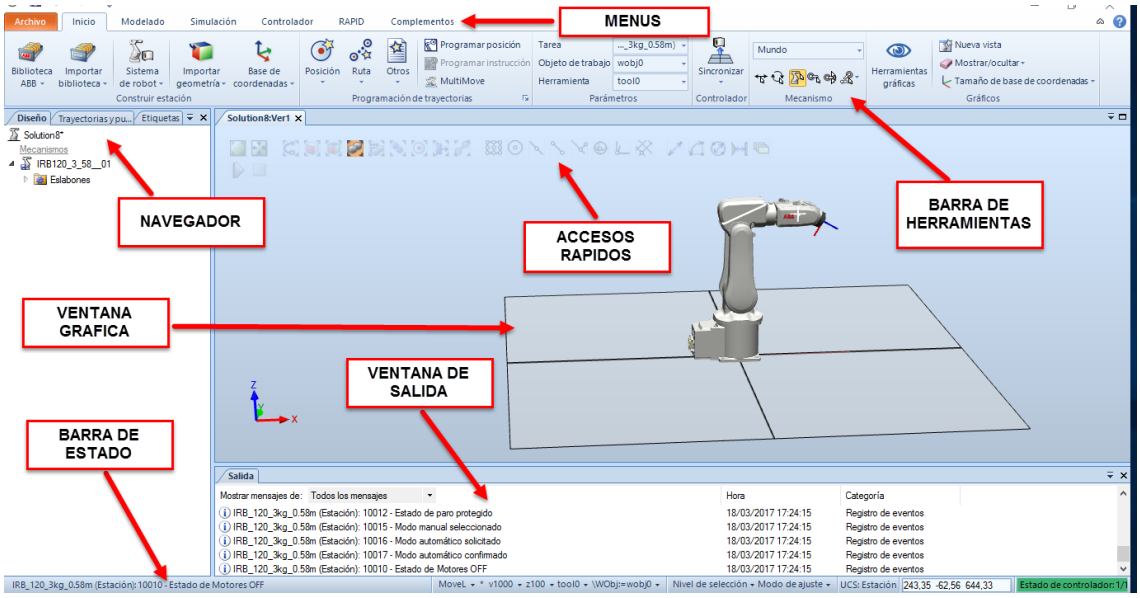


Figura 5-11.Areas del entorno RobotStudio.

- **Menús:** son menús desplegables, que permiten realizar todo tipo de acciones como por ejemplo

- **Barras de herramientas**: dispone de los diversos comandos de las que dispone el programa. Están agrupados por funciones.

- **Ventana gráfica**: en ella se pueden observar los objetos en 3D de la estación, seleccionarlos, etc. Es decir, es la extensión gráfica del navegador. Pueden crearse diversos grupos de pestañas para visualizar a la vez menús.

- **Accesos rápidos de la ventana gráfica:** son accesos rápidos a los comandos del programa más recurridos.

- **Navegador**: es una ventana en la que se muestran de manera jerárquica los objetos de la estación y a su vez se permiten seleccionarlos. Su pueden encontrar dentro de ella otras ventanas diferentes dependiendo del menú en el que se esté.

- **Ventana de salida**: muestra información sobre los eventos que se producen en la estación, por ejemplo cuando se inicia o detiene una simulación. Esta información resulta útil a la hora de solucionar problemas de las estaciones.

- **Barra de estado**: muestra algunos de los parámetros actuales de la estación, como el estado del control virtual, o la posición del sistema de coordenadas del usuario.

5.3 Configuración del entorno de trabajo

El entorno es fácilmente configurable y personalizable. Se pueden cambiar los tamaños de todas las ventanas, o los colores de la ventana gráfica. También se pueden ocultar o mostrar las diferentes ventanas. Incluso existe la posibilidad de abrir varias ventanas gráficas a la vez, que nos ofrezcan diferentes vistas de la estación, o configurar los diferentes aspectos gráficos, las unidades de las magnitudes, los botones de las barras…

Si se está trabajando con más de un menú y se quiere monitorizar a la vez tanto la simulación del robot como la ejecución de las líneas del programa que se está simulando se puede dividir la ventana gráfica en varias pestañas.

Para ellos se debe seleccionar con el botón derecho del ratón la pestaña de la ventana gráfica “nuevo grupo de pestañas”.

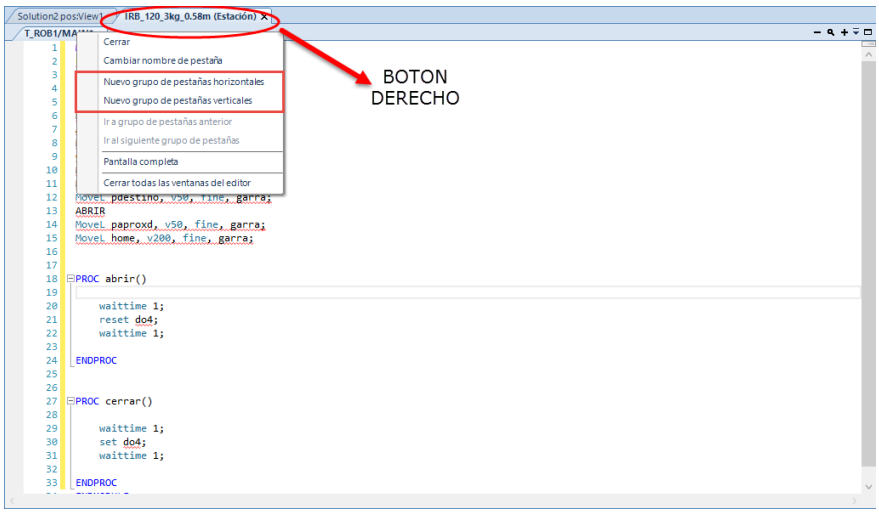


Figura 5-12.Configuración de multiples ventanas gráficas.

Como se pude observar en la imagen anterior, hay dos tipos de grupo de pestañas:

- Nuevo grupo de pestañas horizontales
- Nuevo grupo de pestañas verticales

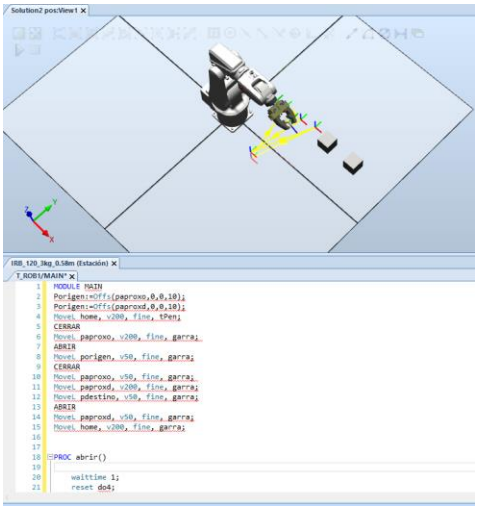


Figura 5-13. Grupo de ventanas horizontales.

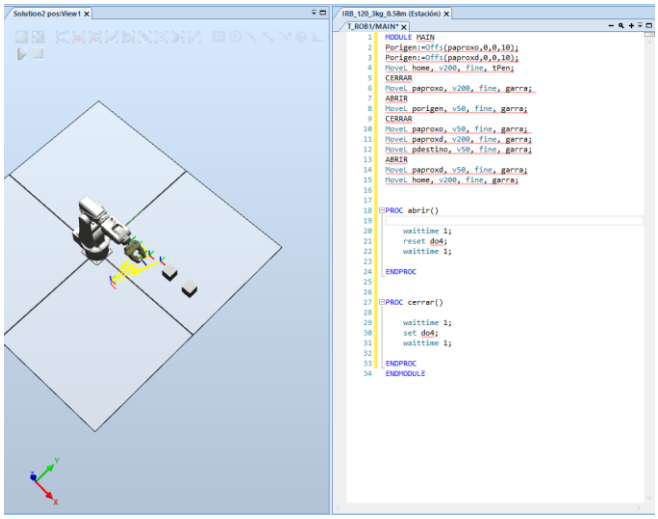


Figura 5-14. Grupo de ventanas verticales.

Para volver a la situación inicial de las pestañas y volver a verlas de una en una se volverá a pulsar el botón derecho del ratón encima de una de las pestañas y se seleccionará la opción “ir al siguiente grupo de pestañas”

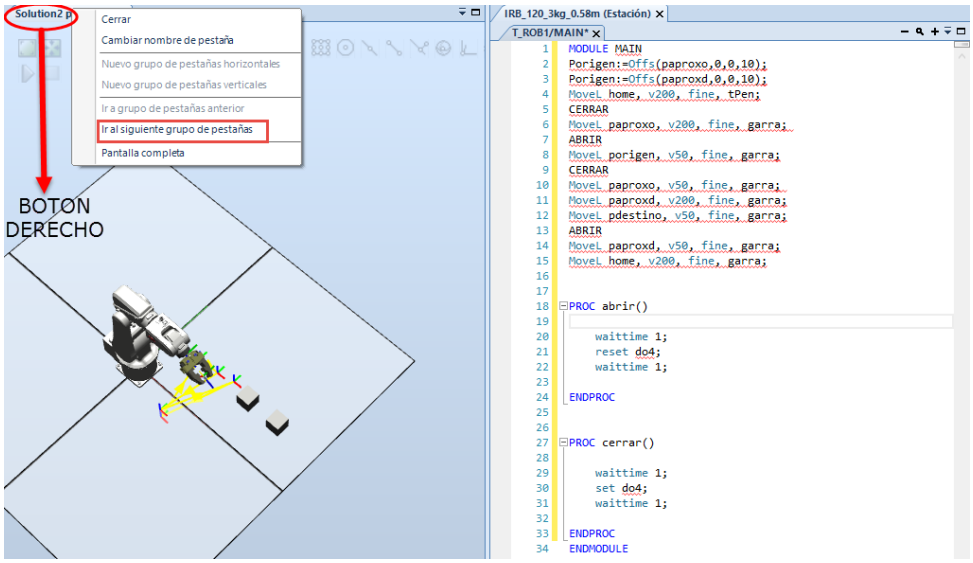


Figura 5-15.Configuración para trabajar con una sola ventana.

5.4 Creación de una nueva estación de trabajo

Al abrir por primera vez el programa aparecerá una ventana similar a la de la figura siguiente, a no ser, que la opción "mostrar siempre esta página en la puesta en marcha" no esté activada. En caso de no estas activada podremos acceder a las mismas opciones a través de la opción *Archivo* de la barra de menú.

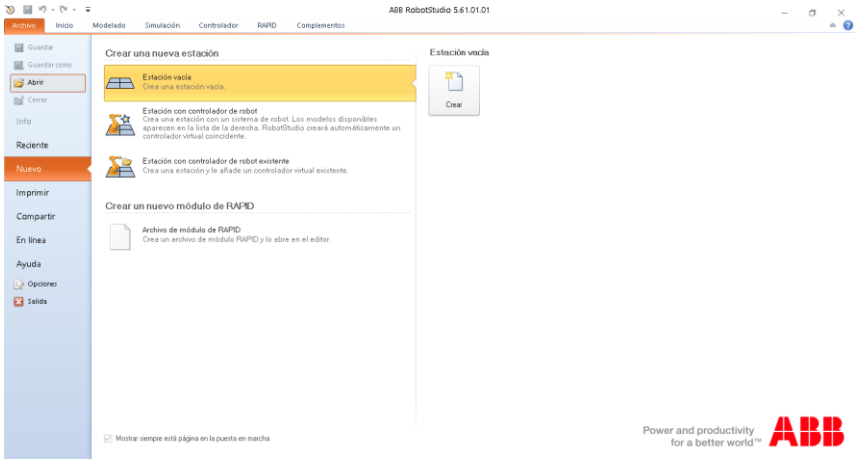


Figura 5-16.Opciones del menú Archivo.

Para comenzar a trabajar será necesario escoger una de las opciones que aparecen en la pestaña ARCHIVO:

1. **Abrir**: Abrir una estación ya existente para seguir trabajando en ella.

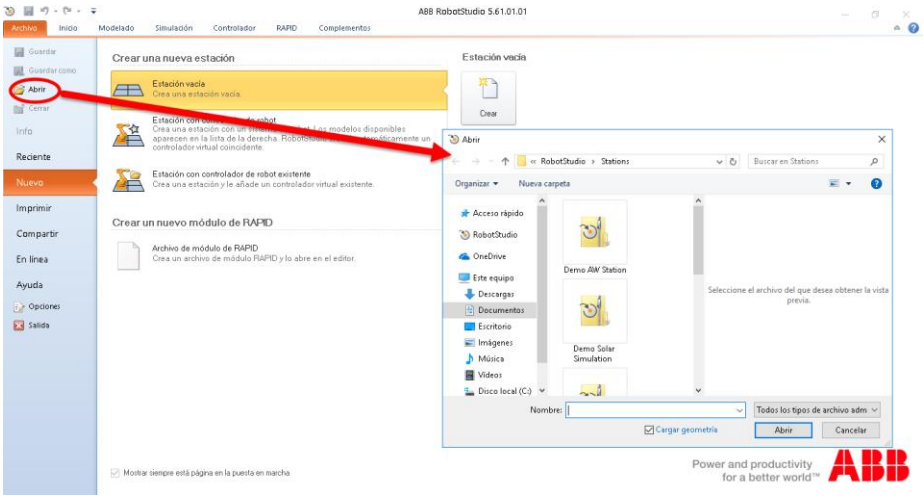


Figura 5-17.Abrir celula de trabajo existente.

2. **<u>Nuevo:</u>** para crear una estación nueva hay tres formas de comenzar.

   - <u>Estación vacía:</u> crea una estación vacía para poder diseñar la celula de trabajo al gusto del usuario.

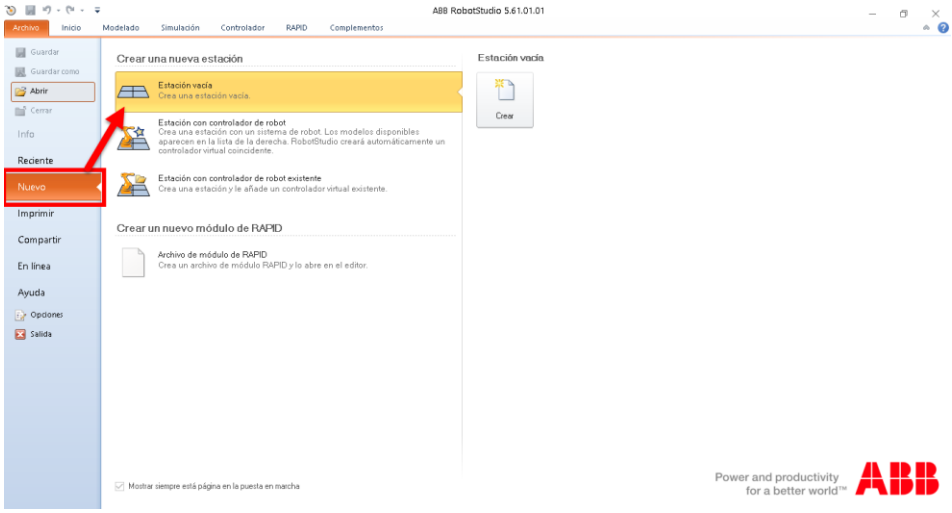


Figura 5-18.Solución nueva con estación vacia.

   - <u>Solución con estación y controlador de robot:</u> crea una solución que contiene una estación y un controlador de robot. Da opción a elegir el manipulador deseado de los existentes en la librería del programa.

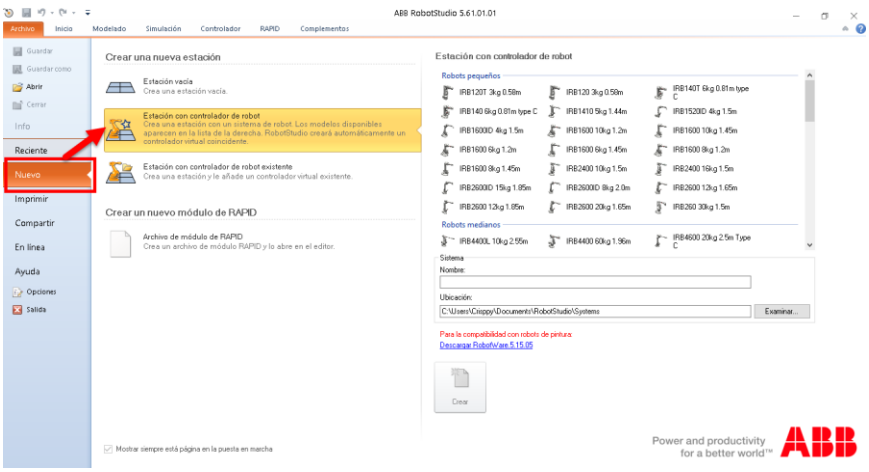


Figura 5-19. Solución nueva con estación y controlador.

- Solución con estación vacía : crea una estructura de archivos de solución con una estación vacía

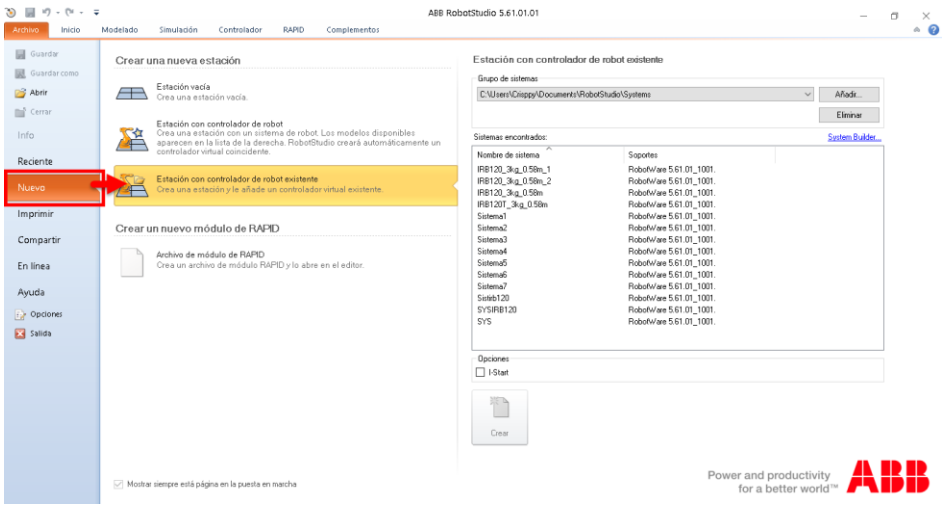


Figura 5-20.Nueva Estación vacía.

Siempre que se sepa cuál es el robot y el controlador con el que se va a trabajar y dichos componentes se encuentren en las librerías del sistema se elegirá la opción "solución con estación y controlador de robot"

5.5 Menú Inicio

El menú Inicio se divide en 6 grupos diferentes que acogen diversas funciones:

- **Construir estación:** contempla funciones específicas para añadir nuevos componentes a la célula de trabajo y nuevas bases de coordenadas con las que trabajar.
- **Programación de trayectorias**: grupo de funciones que permite la creación de objetos de trabajo, trayectorias y posiciones…
- **Parámetros**: selección de manipulador, del objeto de trabajo y de la herramienta de referencia para trabajar.
- **Controlador:** modos de sincronización de la programación con los objetos de trabajo.
- **Mecanismo**: permite realizar movimientos y giros del manipulador tanto en conjunto como eje a eje de forma manual.

- **Gráficos:** son funciones que permiten controlar la vista de los gráficos y modificar su apariencia, añadir nueva ventana de gráficos, modificar el tamaño de la base de coordenadas y la visibilidad de distintos elementos.

5.5.1 Sincronización de la célula de trabajo

Las dos partes, requieren estar "sincronizadas" para se entiendan mutuamente. Esta sincronización se realiza de dos formas diferentes:

- **Sincronizar con RAPID:** para sincronizar objetos de la estación con un código de la estación
- **Sincronizar con estación :** para sincronizar un código de RAPID con objetos de la estación

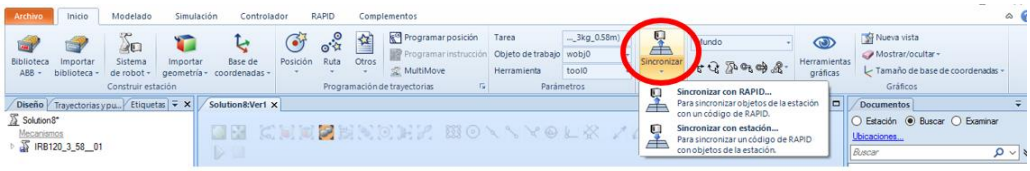


Figura 5-21.Tipos de Sincronización de la célula de trabajo.

5.5.2 Creación de planos de trabajo

Para crear el plano u objeto de trabajo se debe seleccionar en el menú Inicio /Otros/ Crear objeto de trabajo

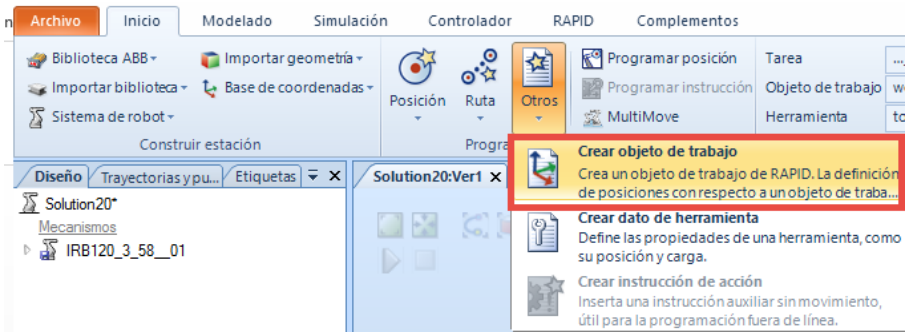


Figura 5-22.Creación de objeto de trabajo.

Es necesario dotar al objeto de trabajo de un nombre y además posicionar su sistema de coordenadas, por ejemplo en la esquina superior. Una vez fijado los parámetros necesarios se aceptará y se dará a crear.

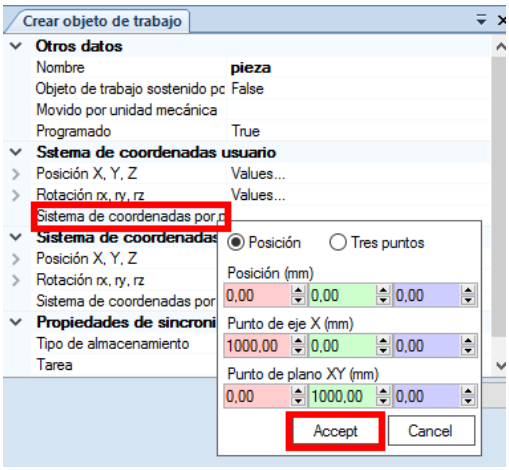


Figura 5-23. Definición sistema de Coordenadas de un nuevo objeto de trabajo.

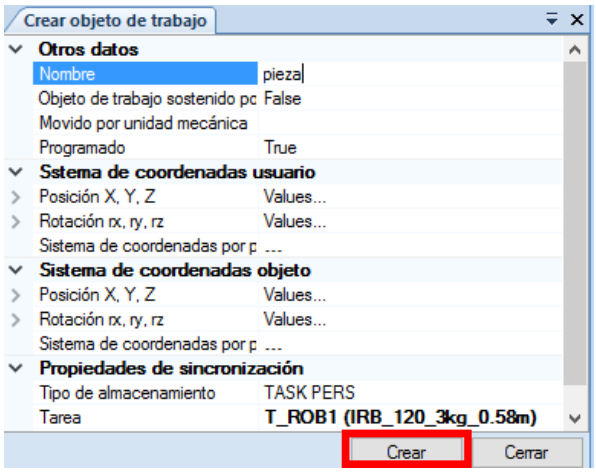


Figura 5-24. Creación de un nuevo objeto de trabajo.

5.5.3 Creación de trayectorias y posiciones

Para poder realizar operaciones y crear posiciones hay que hacerlo respecto a una herramienta de trabajo del robot.

Si se quiere crear posiciones relativas se debe seleccionar la función Posición/ Crear Punto.

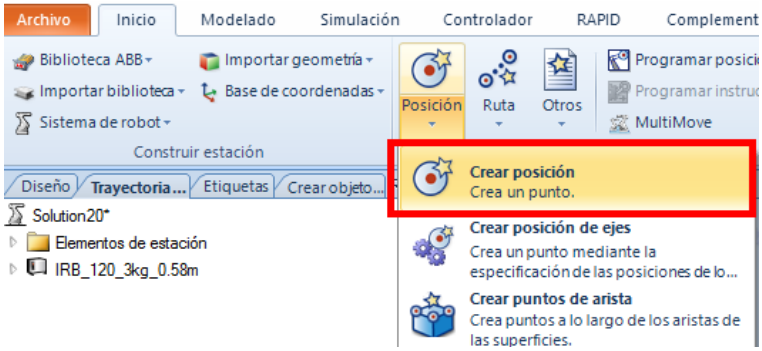


Figura 5-25.Creación de nueva posición.

Se seleccionará el botón *"Añadir"* para poder añadir cada uno de los puntos necesarios para posteriormente generar la trayectoria correspondiente. También se podrá añadir nuevos puntos automáticamente si se realizan movimientos a mano alzada (funciones del grupo mecanismos), si se trabaja con la ventana crear posición cualquier clic con el ratón que se genere en la ventana gráfica será un nuevo punto.

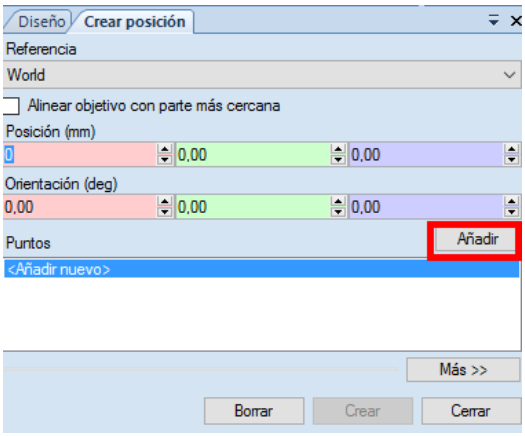


Figura 5-26.Definición de nueva posición.

Se introducirán los parámetros necesarios de posición y orientación deseados, el nombre del punto y el objeto de trabajo al que irá referenciado. Una vez completados dichos campos se procederá a crear el punto seleccionando *"Crear"*.

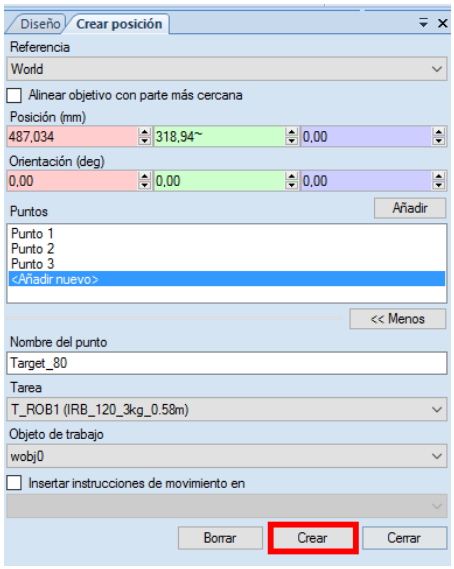


Figura 5-27.Finalización creación de nueva posición.

5.5.4 Toma de medida

Si se quiere realizar una medida en Robotstudio en los accesos rápidos de la ventana gráfica aparecerán una serie de funciones específicas sobre diferentes tipos de medida.

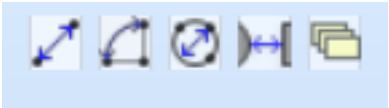


Figura 5-28.Funciones específicas de medida.

Para ser más preciso a la hora de realizar la medida es conveniente usar como apoyo los comandos de referencia forzada.

Figura 5-29.Funciones referencia forzada de medida.

Esta operación de medida de distancias será de gran utilidad para poder verificar las posiciones relativas de los objetos así como la magnitud de las piezas creadas.

5.5.5 Conexión de una herramienta al manipulador

Se puede añadir una herramienta de tres maneras diferentes:

- Desde CAD (ver apartado Modelado)
- Diseñando la propia geometría (ver apartado Modelado)
- Mediante la función Importar biblioteca/ Equipamiento del menú inicio (Herramientas prediseñadas)

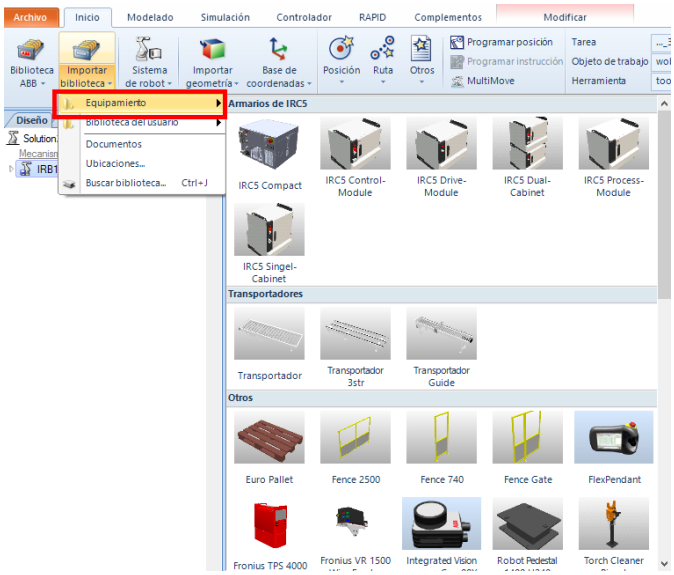


Figura 5-30.Biblioteca de equipamientos en RobotStudio.

Una vez creada la herramienta hay que adjuntarla al robot y posteriormente actualizar su posición de tal manera que se sitúe la herramienta deseada en la muñeca del manipulador. Para ello se arrastrará la herramienta deseada hacia el robot en la ventana del navegador.

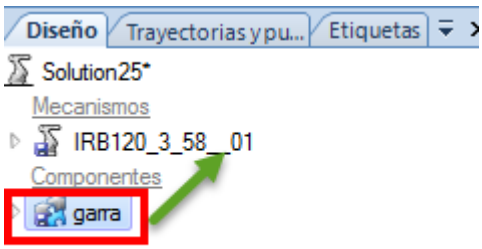


Figura 5-31.Conectar y ubicar la herramienta al robot.

Aparecerá un mensaje preguntando si se desea actualizar la posición de la garra. Seleccionamos SI.

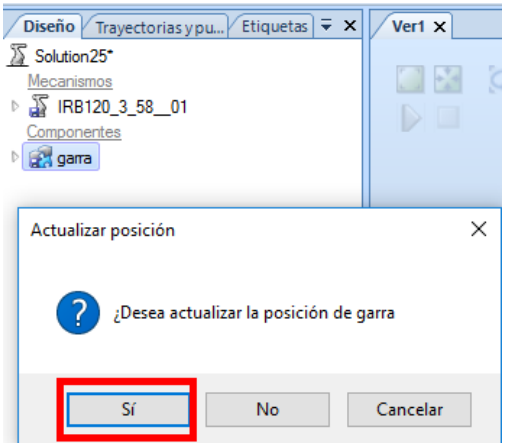


Figura 5-32.Actualización de la posición de la herramienta.

Podremos comprobar que se ha conectado la herramienta al manipulador correctamente cuando se observe la herramienta colocada en la muñeca del robot en la ventana gráfica, además de incluido dentro del elemento 6 del robot.

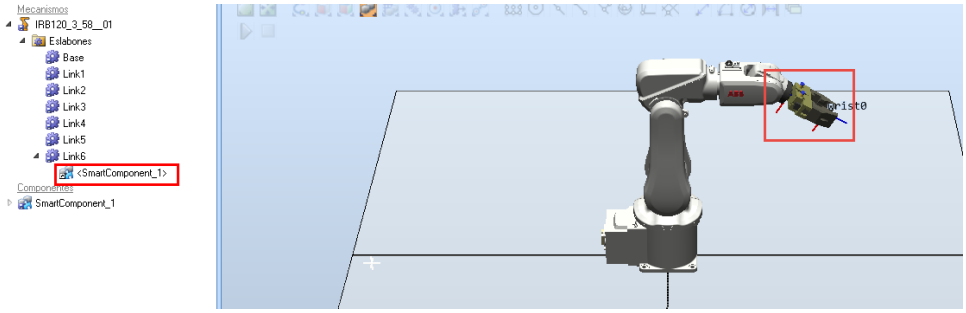


Figura 5-33.Comprobación de la correcta conexión entre herramienta y robot.

Si la herramienta está bien conectada y se mueve el robot con el modo mano alzada, ambos elementos se moverán a la vez como si de uno solo se tratase.

5.6 Menú Modelado

El menú Modelado se divide en 5 grupos diferentes que acogen diversas funciones:

- **Crear:** creación de elementos de la célula de trabajo
- **Operaciones de CAD:** operaciones CAD para aplicar en cuerpos u objetos en 3D
- **Medir:** herramientas de medición
- **Mano alzada:** funciones para realizar diferentes tipos de moviemitno del manipulador de forma manual.
- **Mecanismo:** crear o modificar nuevos mecanismos y crear una herramienta junto con sus datos de herramienta.

5.6.1 Creación de una herramienta nueva

5.6.1.1 Creación de geometría

Existen diversas funciones en el menú simulación para crear una geometría mediante sólidos.

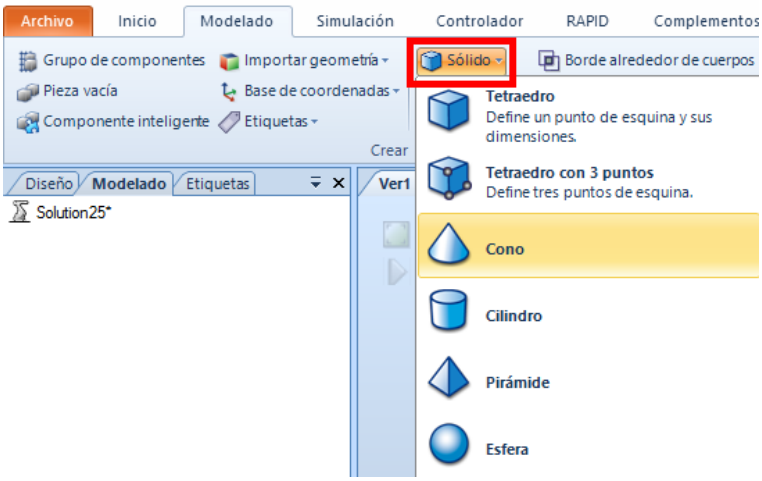


Figura 5-34.Función de geometría mediante solidos.

Además dichas geometrías podrán ser tratadas mediante operaciones de CAD para crear objetos más complejos.

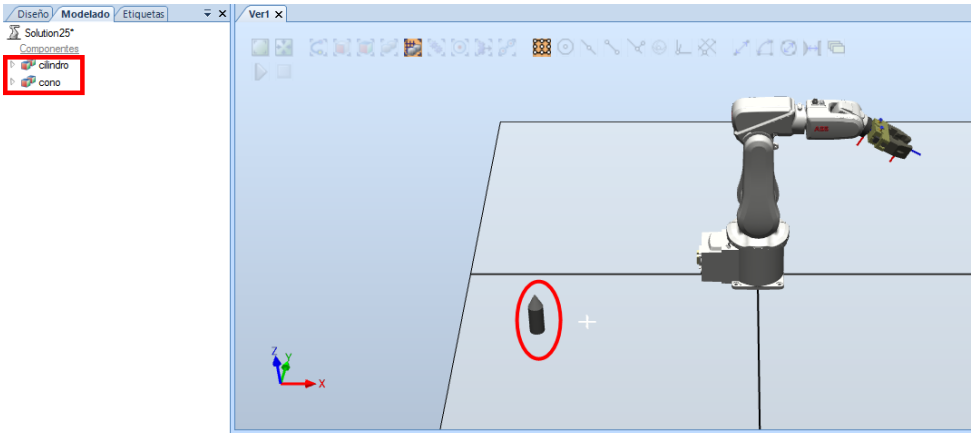


Figura 5-35.Objeto complejo creado por geometrias.

Para unir en una sola pieza el cilindro y el cono usamos la función *"Unión"*

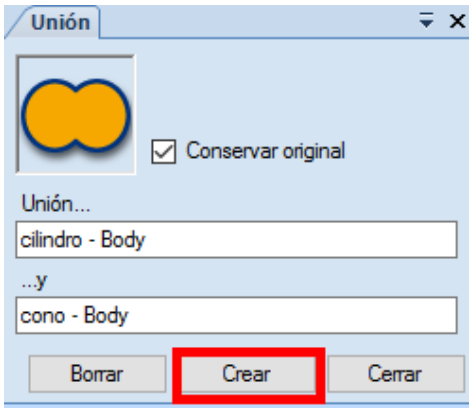


Figura 5-36. Unión de geometrías.

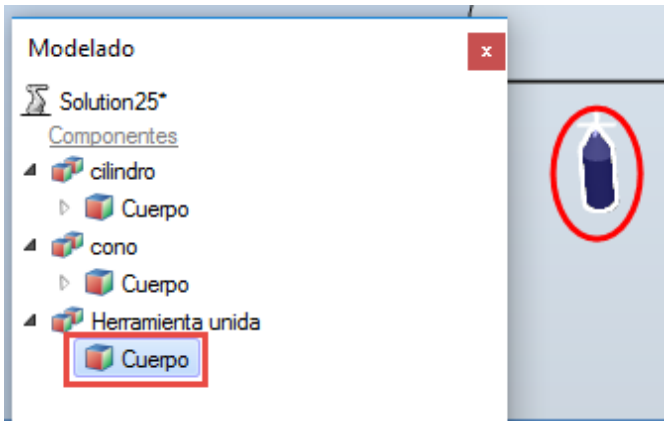


Figura 5-37. Creación de una geometría compleja.

5.6.1.2 Importar geometría

Si la herramienta ha sido diseñada por otro programa manteniendo el formato CAD podrá ser importada a Robotstudio para emplear dicha geometría en nuestra célula de trabajo. Para ello guardaremos los archivos de CAD en la carpeta RobotStudio\ libraries o RobotStudio\ Geometry.

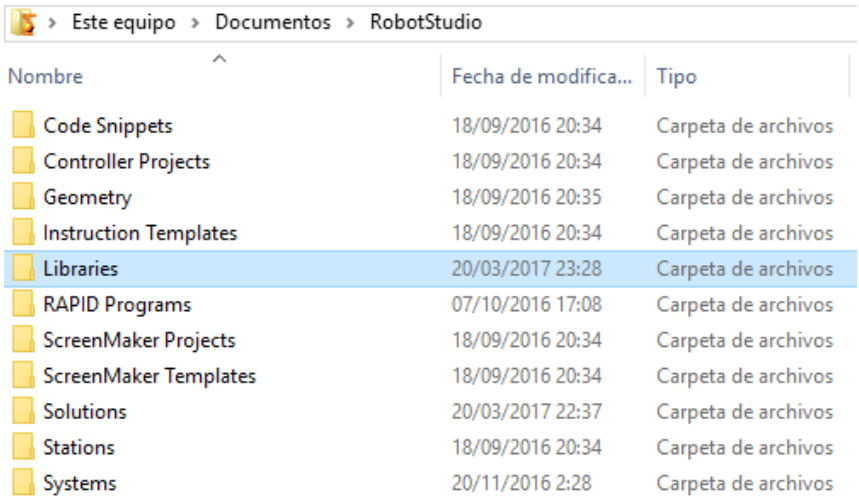


Figura 5-38.Ubicación librerías de RobotStudio.

Para importar la geometría deseada se deberá seleccionar en el menú Modelado *Importar geometría\ Documentos*

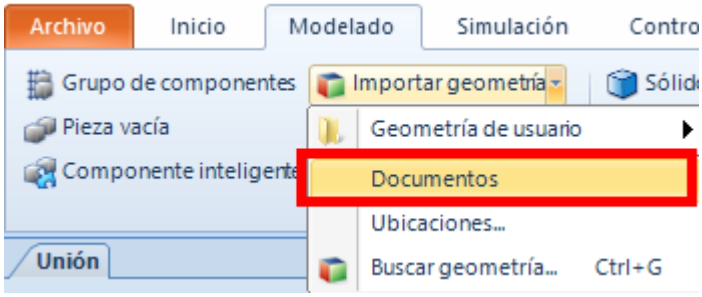


Figura 5-39.Importar una geometría.

Dentro de Biblioteca de Usuario aparecerán los archivos CAD introducidos anteriormente para seleccionar el deseado.

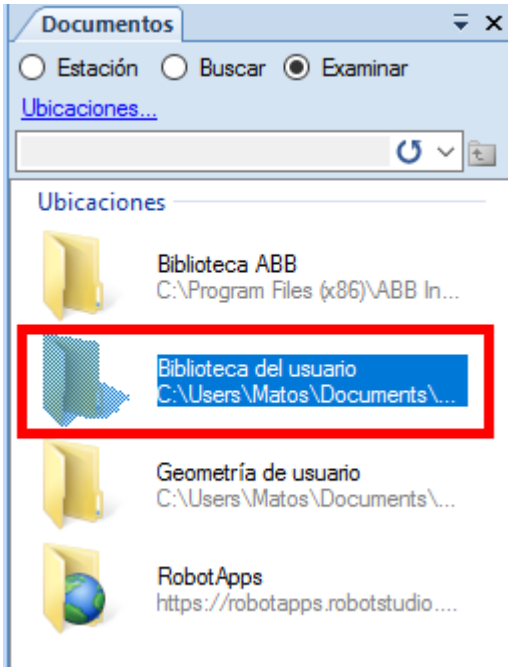


Figura 5-40. Selección de ubicación de la geometría.

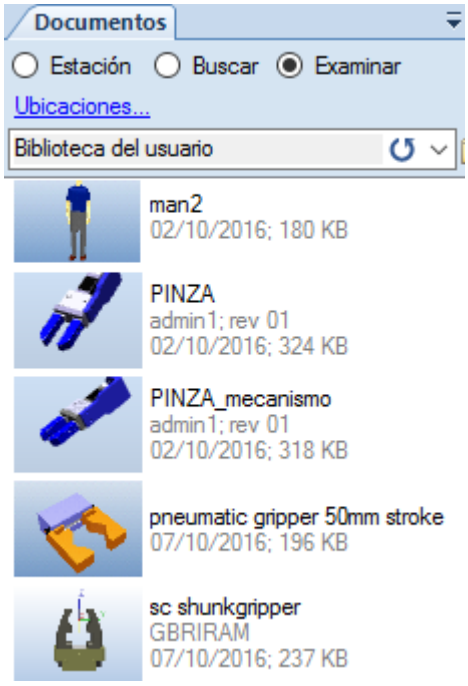


Figura 5-41. Selección de geometría.

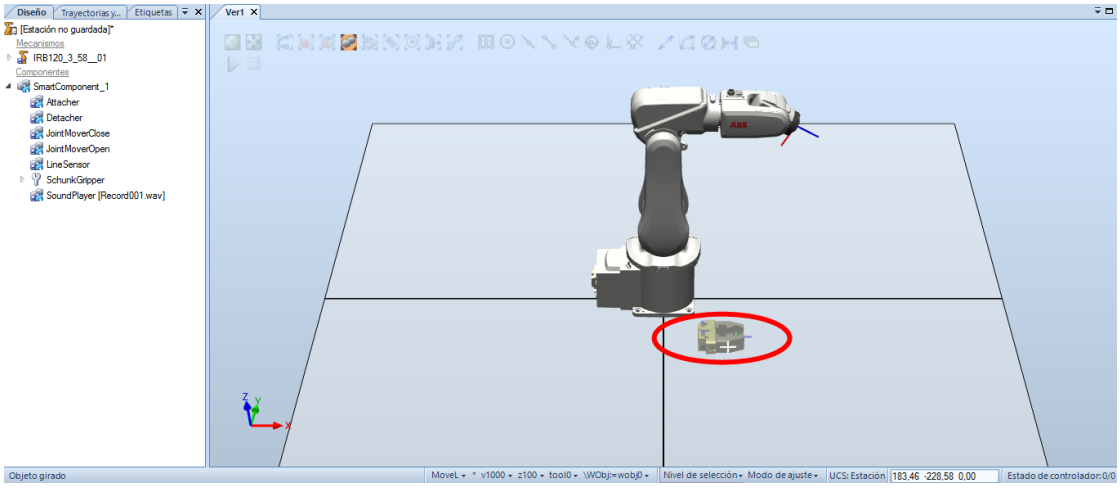


Figura 5-42.Geometría importada al entorno de trabajo.

5.6.1.3 Creación Herramienta

Para transformar en herramienta una geometría creada y que el robot la reconozca como tal será necesario seleccionar la función *"Crear mecanismo"*

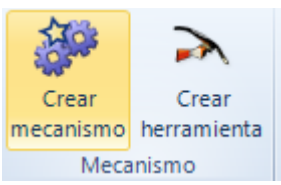


Figura 5-43.Creación de un mecanismo.

Se debe poner un nombre al modelo del mecanismo y seleccionar el tipo de mecanismo (la geometría creada anteriormente).

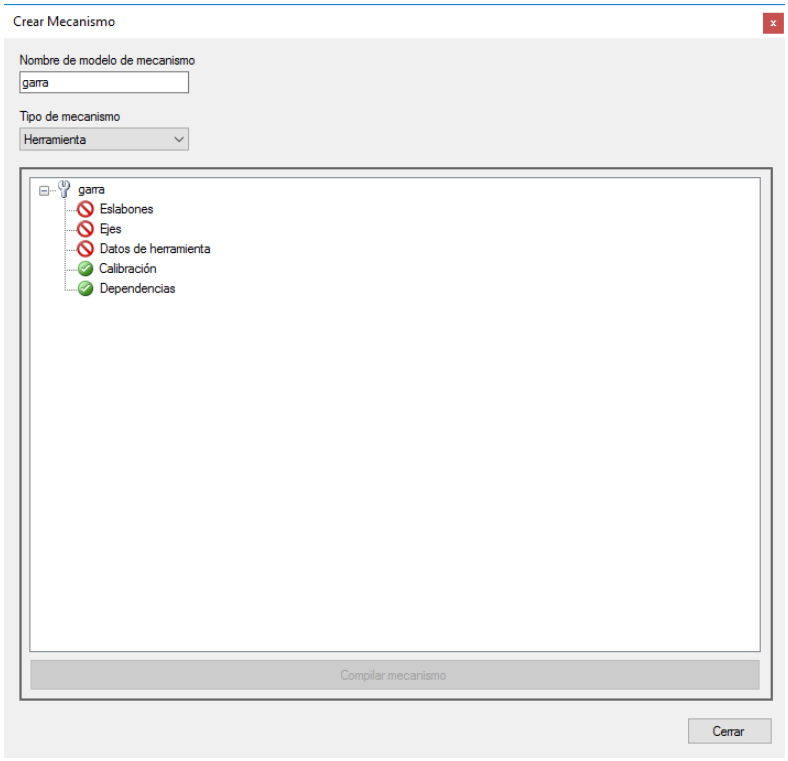


Figura 5-44.Crear una herramienta

Será necesario realizar la configuración de las siguientes especificaciones:

- Crear eslabones: definir los eslabones que compondrán la herramienta

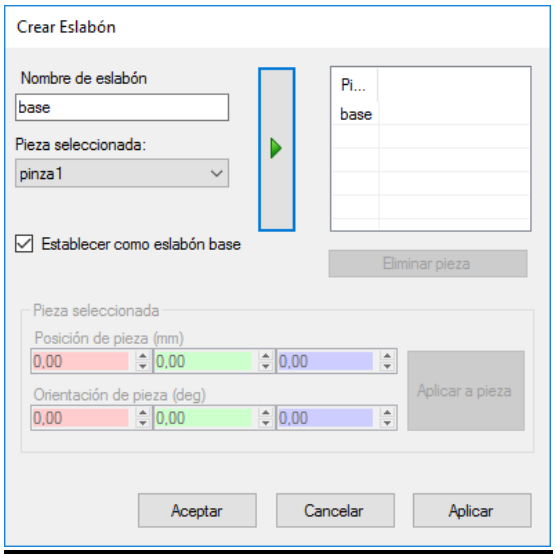


Figura 5-45.Crear eslabones.

- Crear eje: definir los ejes del mecanismo

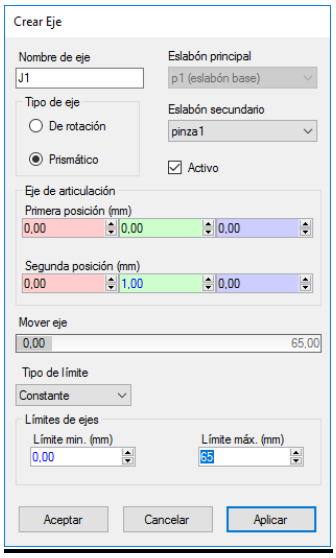


Figura 5-46.Crear ejes.

- Crear datos de la herramienta

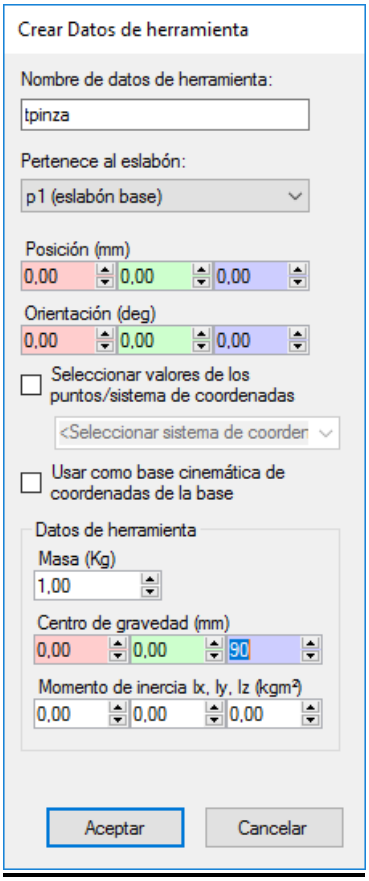


Figura 5-47.Crear datos de la herramienta.

- Crear dependencia: crear al relación de dependencia entre ejes

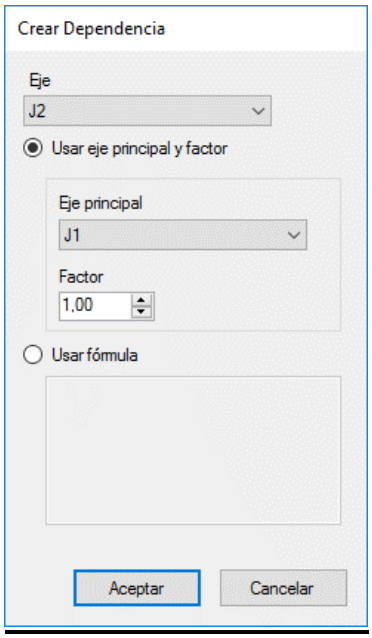


Figura 5-48.Crear dependencia.

Una vez finalizada la configuración de las especificaciones del nuevo mecanismo se procederá a compilarlo aplicando la función "compilar mecanismo".

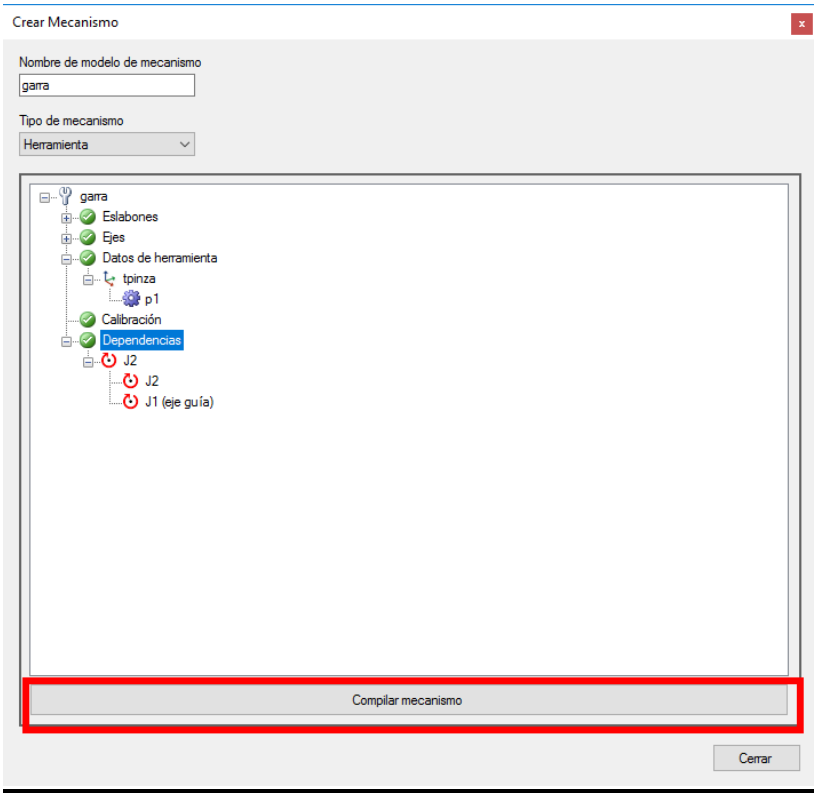


Figura 5-49.Compilar mecanismo.

5.6.2 Creación de componentes inteligentes

Para simular el comportamiento de los distintos elementos de la célula robotizada Robotstudio ofrece la solución de los componentes inteligentes.

Un componente inteligente es un objeto de Robotstudio, con o sin representación gráfica, que representa el comportamiento que puede o desea implementarse mediante la clase code-behind y/o agregación de otros componentes inteligentes. Es decir, son elementos asociados a los sólidos, piezas o robots de la estación, los cuales tienen un comportamiento controlado por señales y propiedades del sistema.

Figura 5-50.Función componente inteligente.

Los componentes modulares básicos que pueden usarse para construir componentes inteligentes son los siguientes:

1. Señales y propiedades: dentro de esta categoría se pueden encontrar elementos para modificar una señal o una propiedad del sistema a programar (puertas lógicas, contadores, temporizadores, expresiones matemáticas, convertidores…)

2. Primitivos paramétricos: Contiene los componentes necesarios para crear de forma automática sólidos y líneas, así como para generar copias de componentes gráficos ya existentes.

3. Sensores: dispone de un conjunto de sensores a utilizar en los procesos de producción automático (sensores de colisión, de línea, de superficie, volumétricos…)

4. Acciones: hace referencia a componentes como conectar o desconectar dos objetos entre sí, eliminar un objeto o simplemente hacerlo visible/invisible.

5. Manipuladores: incluye los componentes necesarios para realizar diversos movimientos, rotaciones y desplazamientos.

6. Otros: contiene representación de una cola de objetos, generación de un número aleatorio, detención de la simulación, reproducción de un sonido, etc.

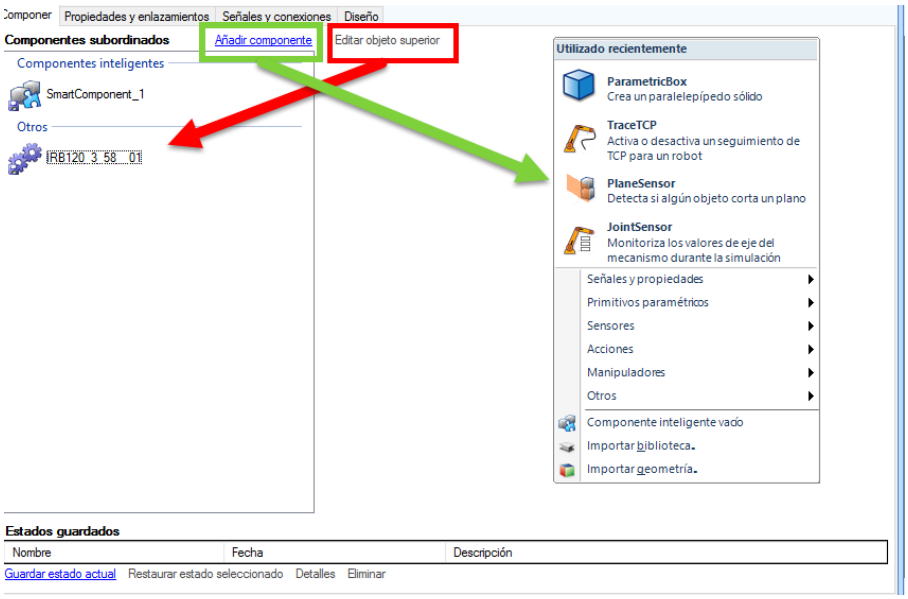


Figura 5-51.Añadir componentes inteligentes.

Se seleccionarán tantos compoenentes como sea necesario para la realización de la lógica de la estación.

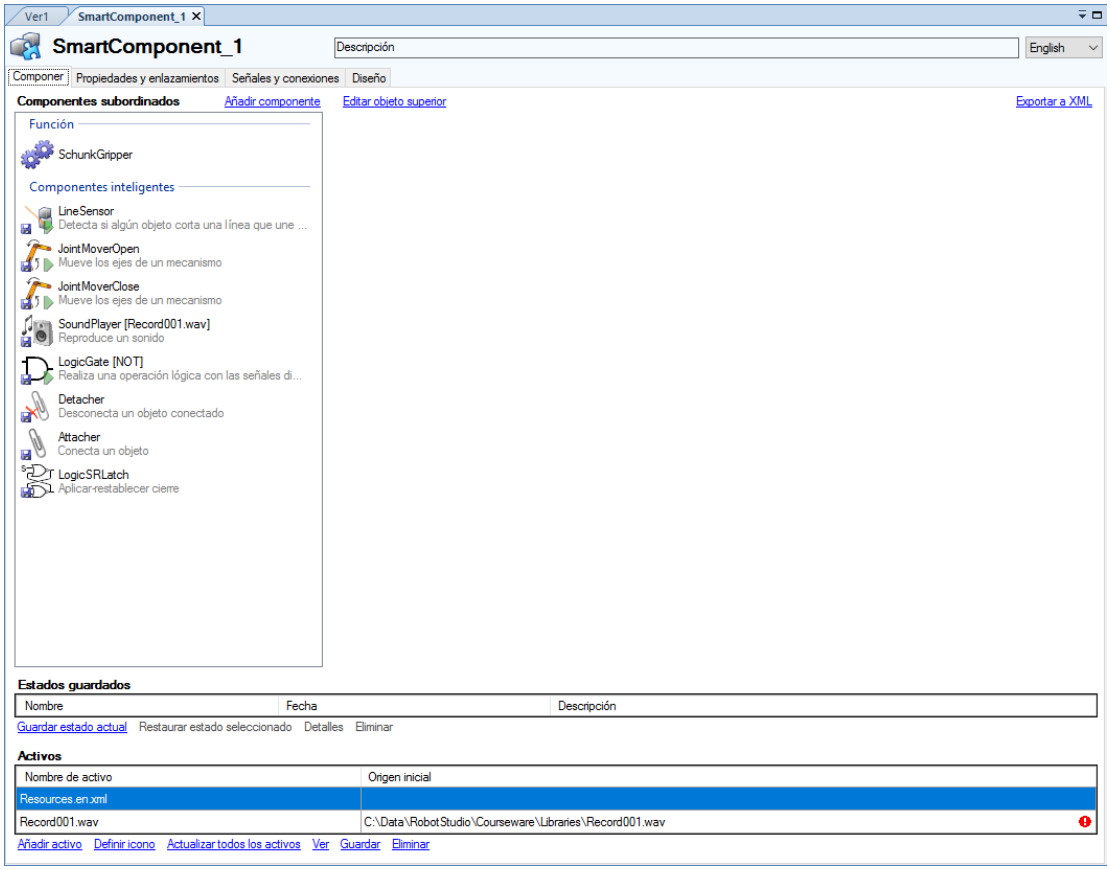


Figura 5-52.Componentes inteligentes de la estación.

En la pestaña “Señales y conexiones” se podrán definir nuevas señales de E/S que se vayan a necesitar a la hora de realizar las conexiones de los componentes inteligentes.

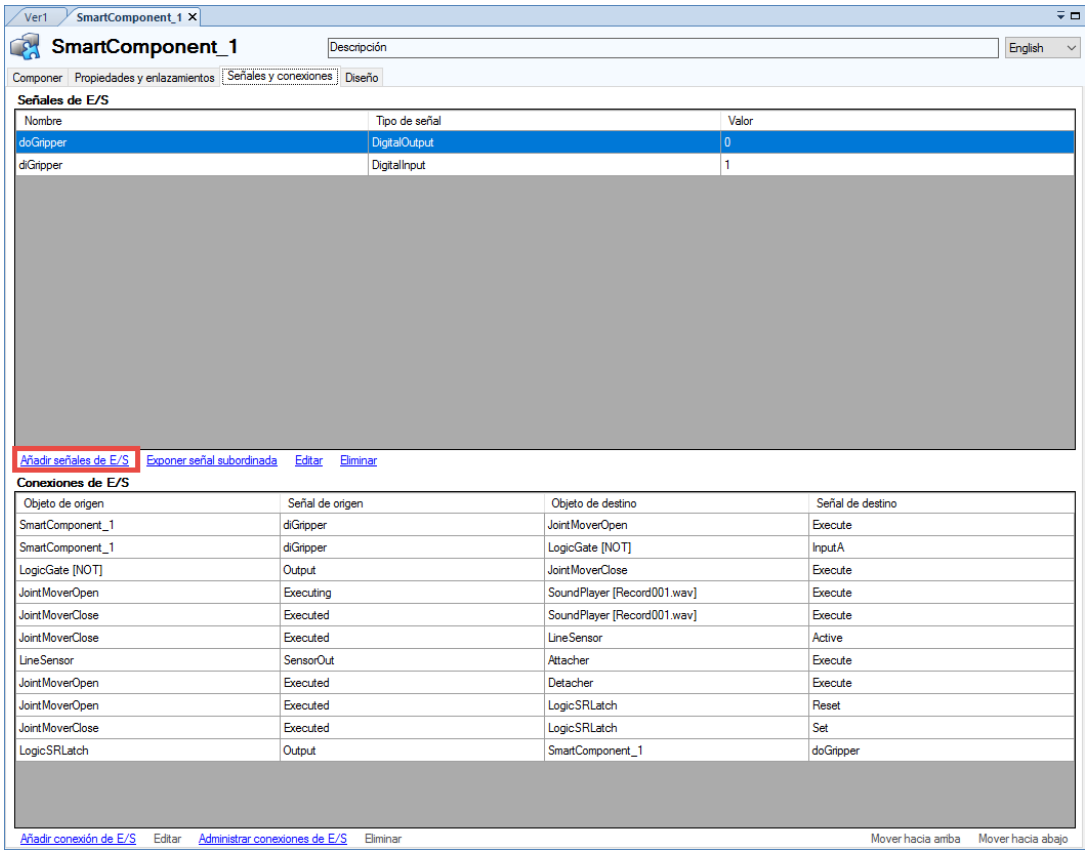


Figura 5-53.Añadir conexiones de E/ S.

5.7 Menú Simulación

El menú Simulación se divide en 7 grupos diferentes que acogen diversas funciones:

- **Colisiones:** configurar la detección de colisiones entre objetos móviles.
- **Configurar:** configuración de la simulación y de la lógica de la estación.
- **Control de simulación:** funciones de inicio, paro, pausa y restablecimiento de la simulación.
- **Monitor:** permite visualizar el estado de las E/S, el cronómetro y la trayectoria que recorre el TCP.
- **Analizador de señales:** configuración, visualización y análisis de los datos de las señales.
- **Grabar película**: graba la simulación, la ventana de gráficos o la ventana de aplicación en una secuencia de video.
- **Conveyor tracking**: para definir la velocidad y la dirección del transportador o reestablecer en transportador a la posición inicial.

5.7.1 Activación del TCP

Para poder visualizar la trayectoria que realiza el TCP cuando se ejecutan una serie de instrucciones de movimiento se debe activar el rastreo de TCP dentro del menú Simulación/Monitor. Se marcará la casilla "*activar rastreo de TCP*".

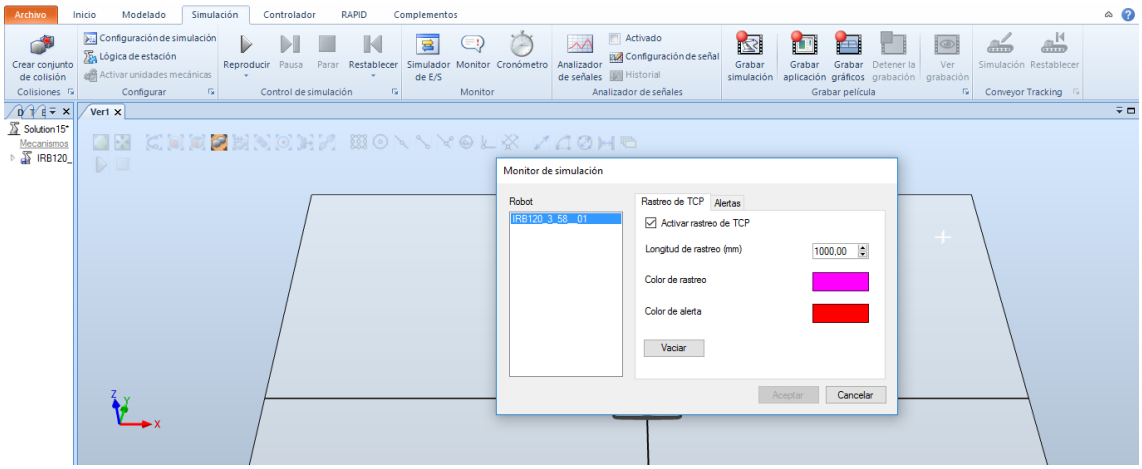


Figura 5-54.Monitor de simulación.

Al ejecutar la simulación se mostrará el recorrido que realiza el TCP de la muñeca o herramienta durante el programa que esté ejecutando.

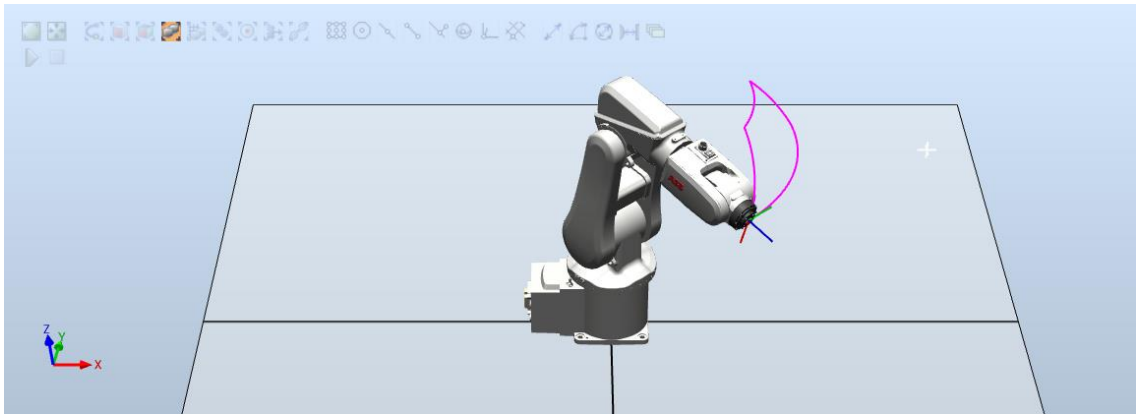


Figura 5-55.Trayectoria que realiza el TCP.

5.7.2 Lógica de la estación

Es necesario interconectar de forma adecuada los componentes inteligentes y el controlador, para que el sistema funcione correctamente coordinando entre sí los movimientos automáticos de los robots y los sólidos inteligentes. Para ello, hay que acceder al menú Simulación y seleccionar *"Lógica de estación"*.

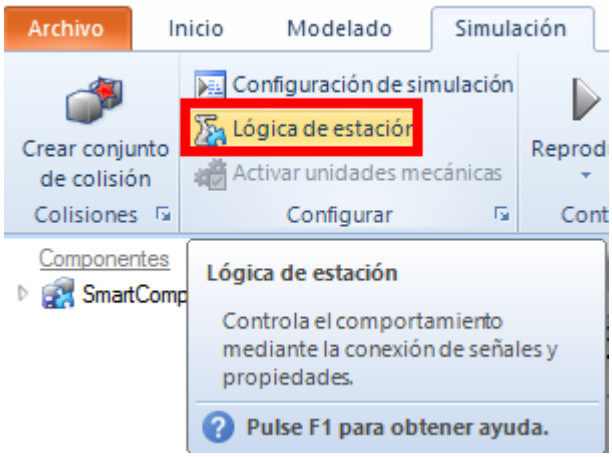


Figura 5-56.Función Lógica de la estación.

La lógica de la estación está formada por un conjunto de bloques correspondientes a los componentes inteligentes y el controlador anteriormente creados. Para poder observar el esquema de bloques una vez abierta la lógica de estación hay que pulsar sobre la pestaña *"Diseño"*.

La conexión entre los bloques es tal que las entradas a los controladores son salidas de los componentes inteligentes y viceversa.

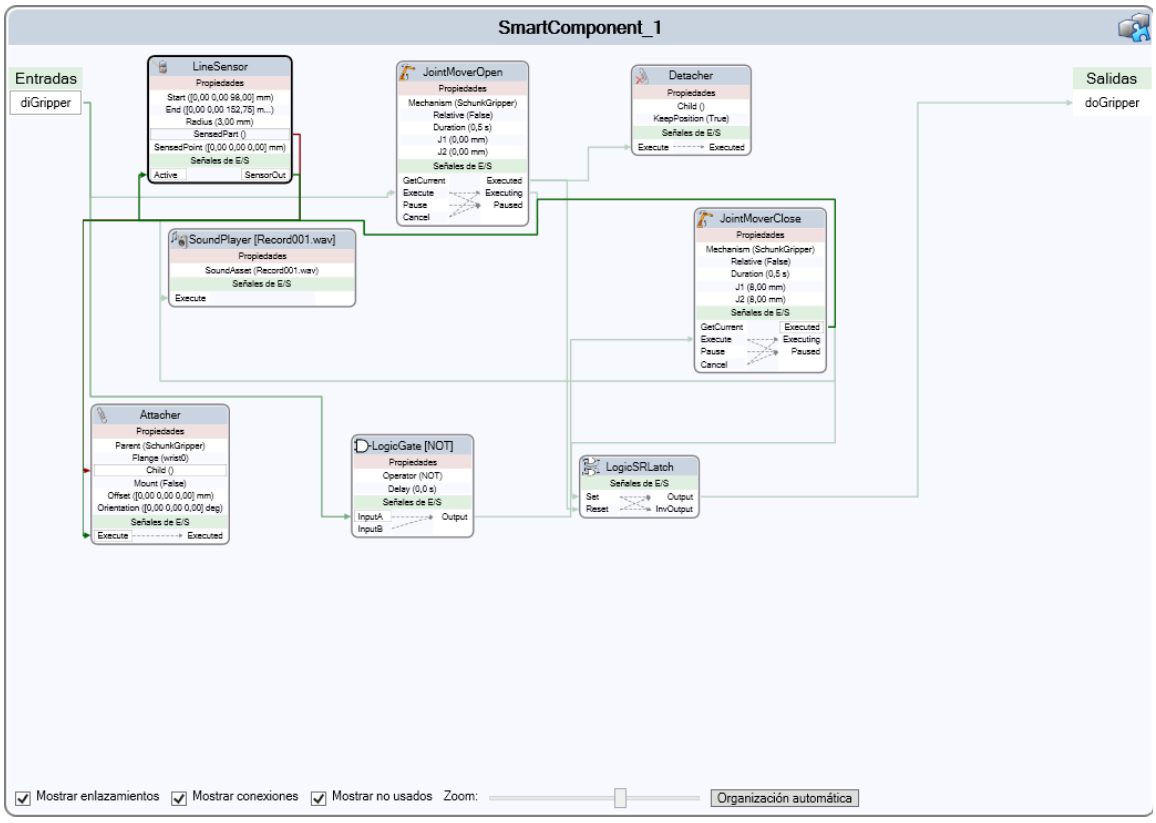


Figura 5-57.Logica de la estación.

5.8 Menú Controlador

El menú Simulación se divide en 5 grupos diferentes que acogen diversas funciones:

- **Acceso :** conexión al puerto de servicio de un controlador físico
- **Herramientas de controladores:** reinicia el controlador y activa los cambios del sistema, crea una copia de seguridad del sistema, muestra las señales de E/S del sistema, el registro de eventos del controlador, virtual flexpendant.
- **Configuración:** permite la configuración de los parámetros del sistema, del controlador, de robotware, del sistema de visión y de los elementos de seguridad.
- **Controlador virtual:** permite el acceso al panel de control, a la ventana del operador, configuración número de funciones especiales en el controlador, definición base de coordenadas de la tarea.
- **Transferir:** para crear un controlador virtual que se corresponda con su sistema y una relación de transferencia con el sistema original.

5.8.1 Conexión de la comunicación con el controlador real

Para poder trabajar dentro de la aplicación RobotStudio con el controlador real de nuestro objeto de estudio será necesario añadirlo al software, para ello se usará el comando *"Añadir controlador"* que se encuentra en la barra de herramientas del menú Controlador. Si el controlador no apareciera en la lista, habría que escribir la dirección IP en el cuadro "Dirección IP" y a continuación actualizar. Una vez aparezca se seleccionará dicho controlador.

Figura 5-58.Añadir controlador.

Al desplegar el menú de opciones de la función *"Añadir controlador"* aparecerán las siguientes formas de añadir un controlador

- **Conexión en un clic:** se conecta al puerto de servicio de un controlador
- **Añadir controlador:** añade los controladores disponibles en la red
- **Iniciar controlador virtual:** para iniciar un controlador virtual y conectarse a él.

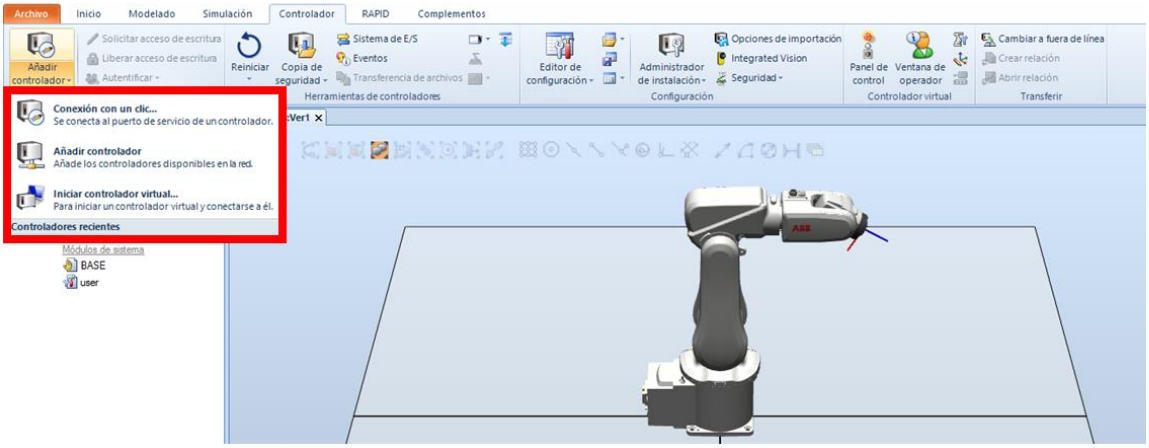


Figura 5-59.Tipos de conexiones con el controlador.

Cuando el robot real se encuentre integrado en el susodicho software se podrá acceder a los controladores del robot. Para poder editar programas y configuraciones, o si se desea hacer cualquier otro cambio en los datos del controlador será necesario pedir el acceso a escritura al robot real, el cual en su Flexpendat aparecerá el permiso para aceptar o denegar.

5.8.2 Activación del uso del virtual flexpendant

Este menú te permite disponer de manera virtual de un cuadro de control como el que se encuentra en la unidad de potencia y control IRC5 del propio robot real. También te permite simular un flexpendant que realizará exactamente las mismas funciones que el real y se manejará de la misma manera.

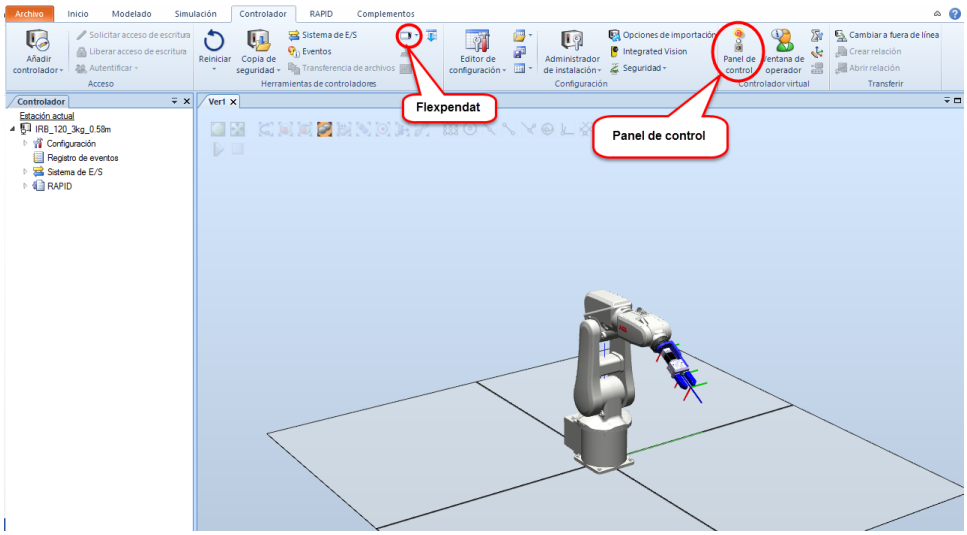


Figura 5-60. Funciones del menú controlador .

Para poder usar el Flexpendant primero será necesario acceder al "*Panel de Control"* y modificar el selector de modo de trabajo, que viene predeterminado en automático, a modo manual (el seleccionado aparece marcado con un puno negro en el círculo)

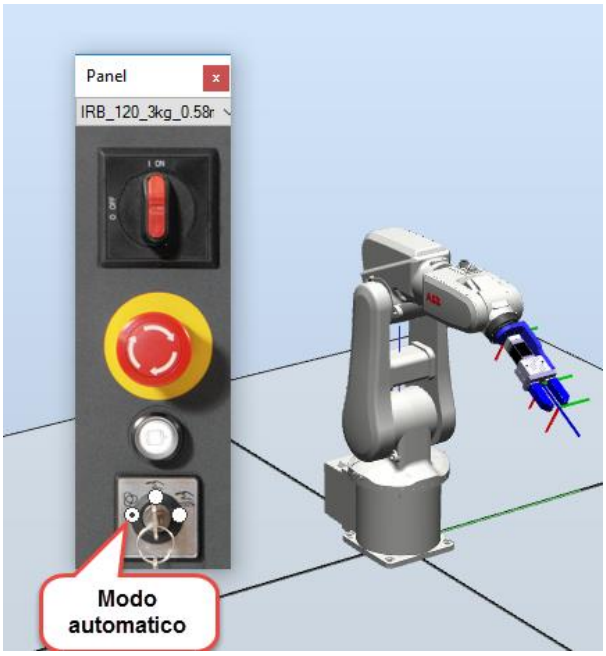


Figura 5-61. Modo de trabajo automatico

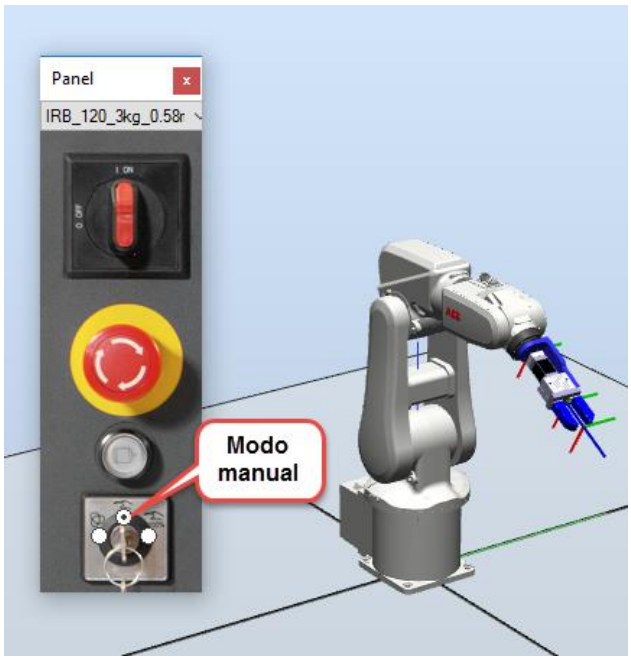


Figura 5-62. Modo de trabajo manual .

Una vez realizado el cambio de modo de modo de trabajo se podrá trabajar con el flexpendant, para ello se seleccionará el comando de dicho dispositivo que se encuentra en el menú Controlador/ Virtual Flexpendant.

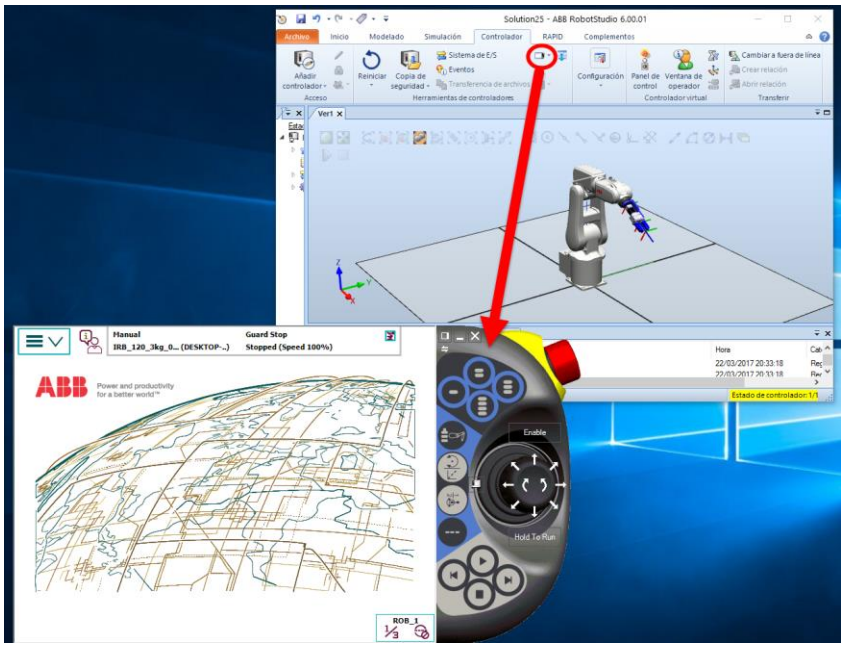


Figura 5-63. Apertura del Flexpendant.

5.8.3 Manejo del virtual flexpendant

En el flexpendant virtual, al igual que el real, dispone de los siguientes accesos:

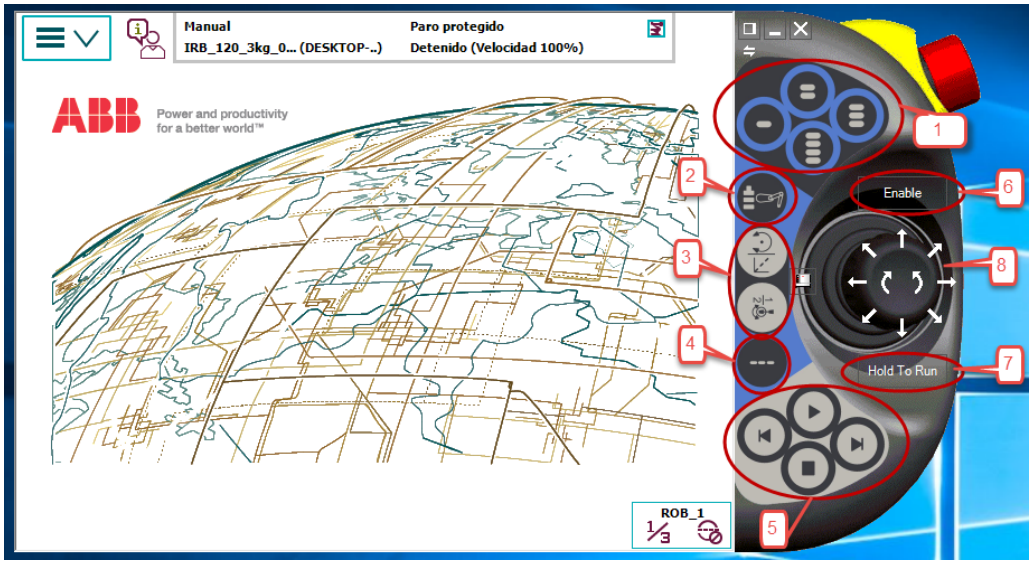


Figura 5-64. Accesos del Flexpendant .

1. Botones programables para llamas a diferentes funciones.
2. Seleccionar el robot al que nos dirigimos
3. Tipo de movimientos que se pueden realizar con el joystick
   - primer botón movimiento eje a eje, movimiento lineal , reorientación
   - segundo botón cambio de ejes en el movimiento eje a eje
4. Tipo de incremente
5. Ejecución del programa
6. Pulsador de habilitación en el lateral

7. Hold to run : cuando ejecutas el programa a modo de prueba hay que pulsar la flecha play continuamente, este botón simula esa acción
8. Joystick de tres direcciones

Si se despliega el icono del listado  se encontrarán una serie de opciones a través de las cuales se trabajará en el proceso.

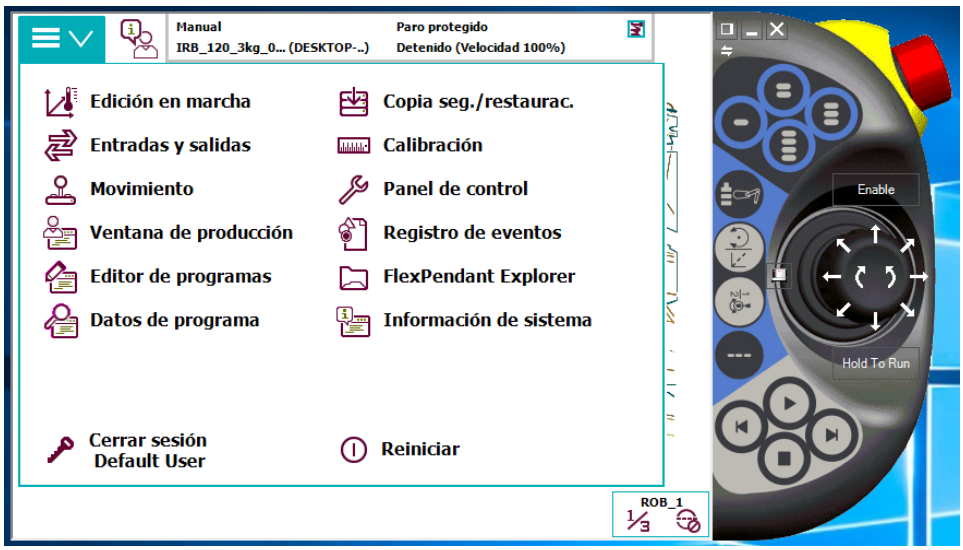


Figura 5-65. Opciones de trabajo del Flexpendant .

5.8.3.1 Menú Movimiento

Para realizar movimientos de manera manual con el manipulador se empleará la función menú en la cual encontraremos los siguientes parámetros que se podrán modificar según la necesidad:

- **Unidad mecánica:** permitirá seleccionar la unidad mecánica con la que vamos a trabajar
- **Modo movimiento :**
  - Movimiento eje a eje : mover los ejes del robot uno a uno
    - Ejes 1, 2 , 3
    - Ejes 4, 5 ,6
  - Movimiento lineal: mover el robot a través de las trayectorias del sistema de coordenadas.
  - Reorientación
- **Sistema de coordenada :** en el movimiento lineal se podrá seleccionar el sistema de coordenadas que se tomará de referencia para realizar el movimiento
- **Herramienta :** se podrá seleccionar la herramienta definida anteriormente que se ajuste a nuestras necesidades
- **Objeto de trabajo :**se podrá seleccionar el objeto de trabajo definido anteriormente
- **Carga útil :**se podrá seleccionar la carga útil definida anteriormente
- **Bloque joystick :** Bloquear para que no actúe el joystick en alguno de los sentidos
- **Incremento:** limitar cuanto se va a mover el robot cuando das un toque en el joystick inferior a un segundo
  - Ninguno
  - Pequeño : 0.01mm
  - Mediano : 0.2 mm
  - Grande: 1 mm
- **Formato de precisión:** nos permite elegir si deseamos ver la dirección en ángulos de Euler o cuaternios

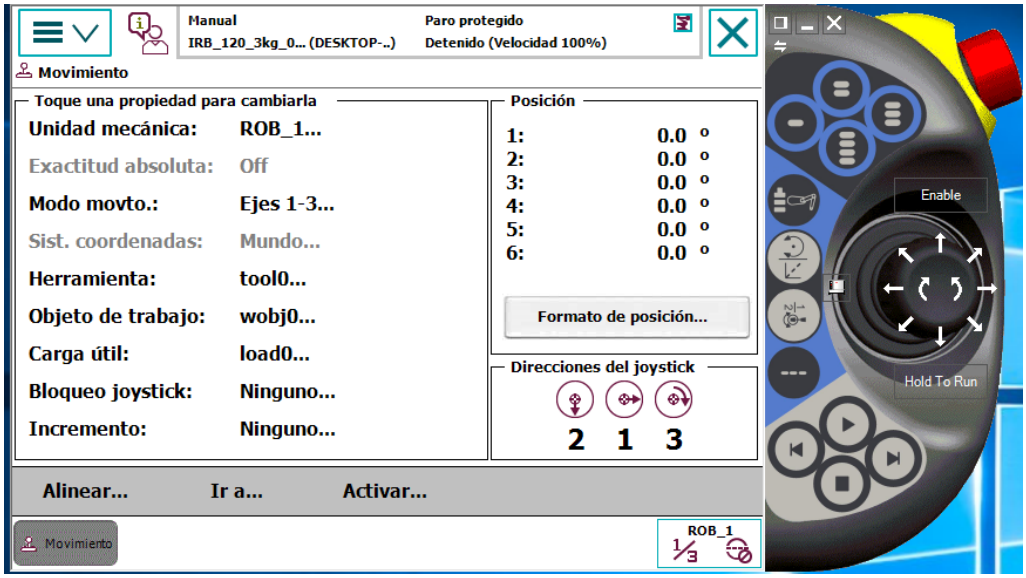


Figura 5-66. Función movimiento.

5.8.3.2 Menú Ventana de producción

Se emplea para cargar un programa desde una unidad en red o desde un usb

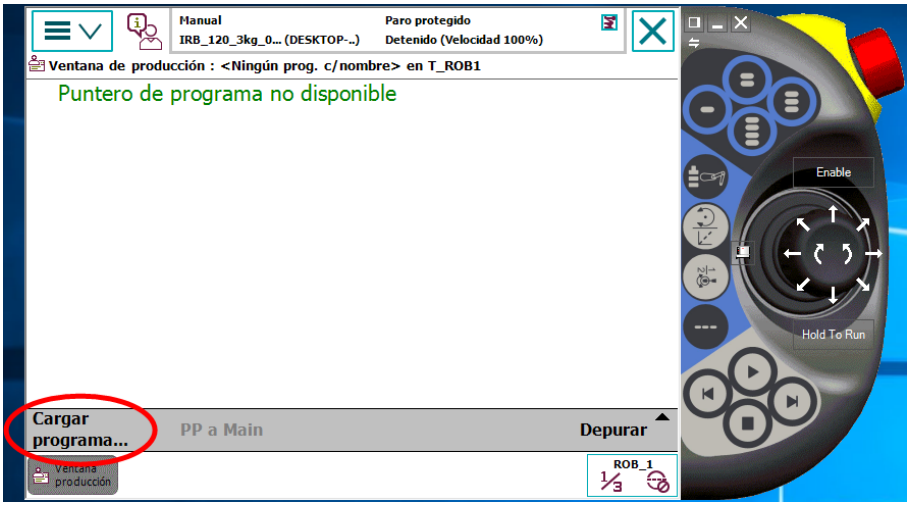


Figura 5-67. Cargar programas desde Usb o red.

5.8.3.3 Menú Editor de programa

Sirve para crear o editar instrucciones de un programa. Esta función nos permite visualizar y modificar el programa a ejecutar o crear un nuevo programa e ir introduciendo las instrucciones y rutinas necesarias.

Al igual que en el editor de programación del software robot estudio desde el flexpendant también se puede programar las líneas de código para realizar el proceso que se desea. Lo primero de todo es crear un programa nuevo o cargar uno existente, en este caso se creará uno nuevo.

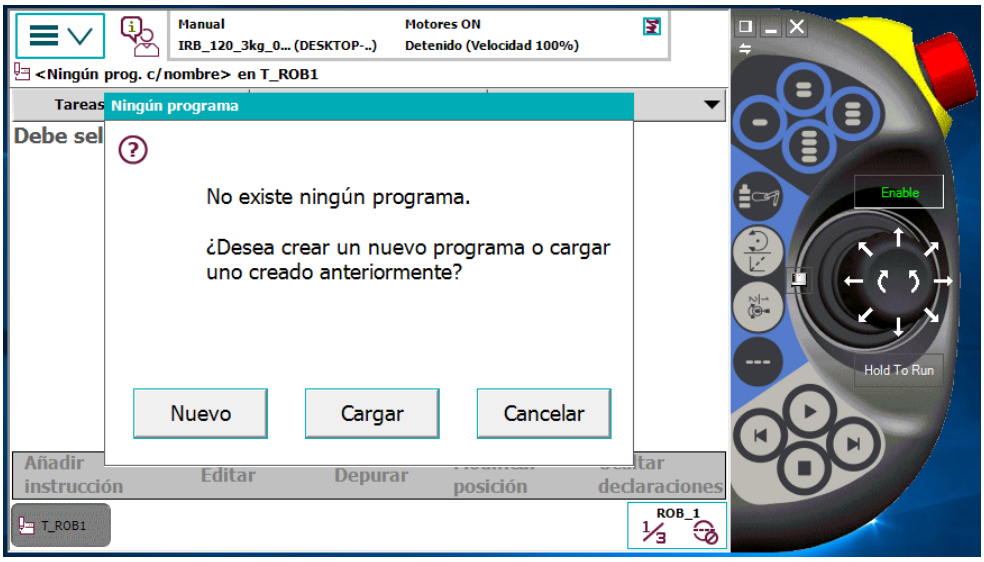


Figura 5-68. Crear o Cargar un programa.

Automáticamente al seleccionar la opción "*nuevo*" se creará un procedimiento denominado "main" que se trata del módulo principal. En caso de querer trabajar con varias rutinas a las que se llamen desde el módulo main se deberá seleccionar el menú "*Rutinas*".

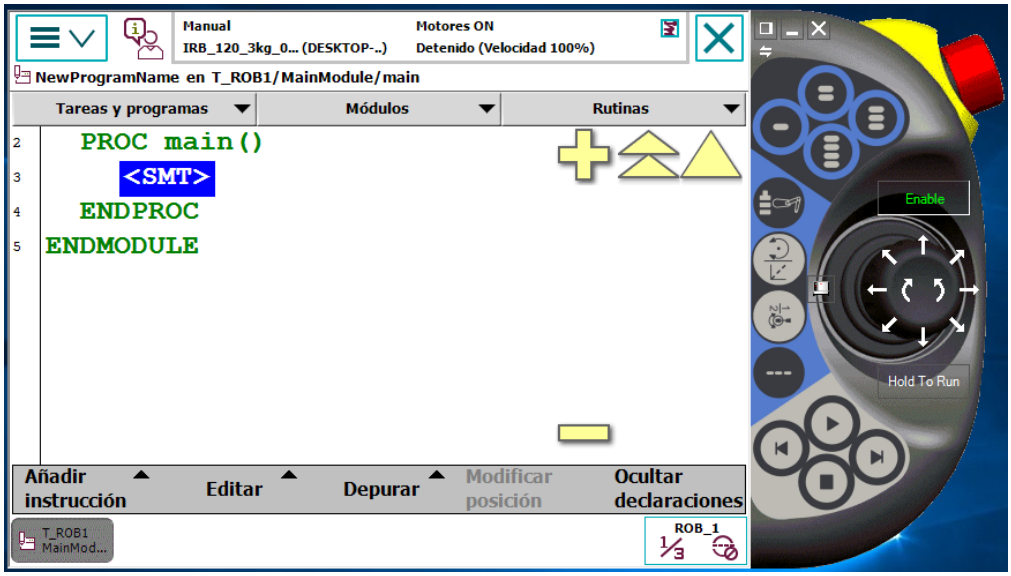


Figura 5-69. Ventana de programación creada.

Una ven aparezca la ventana de rutinas para añadir una nueva rutina se debe seleccionar el botón de "Archivo", "nueva rutina…"

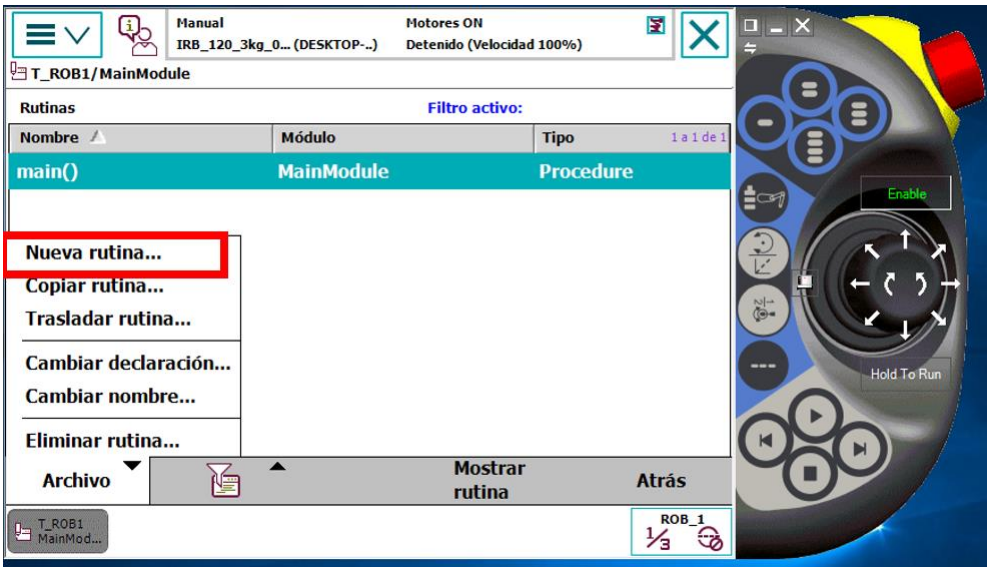


Figura 5-70. Introducción de nueva rutina.

Desde esta pestaña de Rutinas se podrá seleccionar la rutina con al que se quiera trabajar a la hora de introducir o modificar las líneas de código necesarias para la aplicación correspondiente.

Una vez seleccionada la Rutina correspondiente para comenzar a introducir instrucciones en el botón "añadir instrucciones"

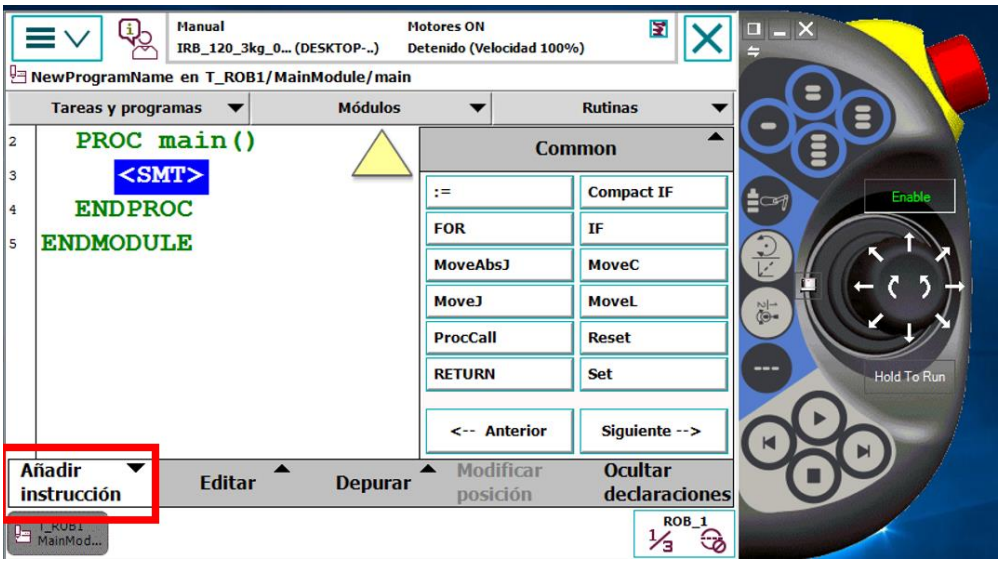


Figura 5-71. Añadir instrucción.

Al introducir una instrucción te viene por defecto una serie de parámetros que se podrán modificar, en cambio otros son necesarios definirlo, como por ejemplo la posición (como se ven en la imagen siguiente aparece la posición marcada con *).Para seleccionar la posición si hay puntos ya registrados en el controlador se podrán utilizar y sino se podrá crear uno nuevo seleccionando la opción nuevo

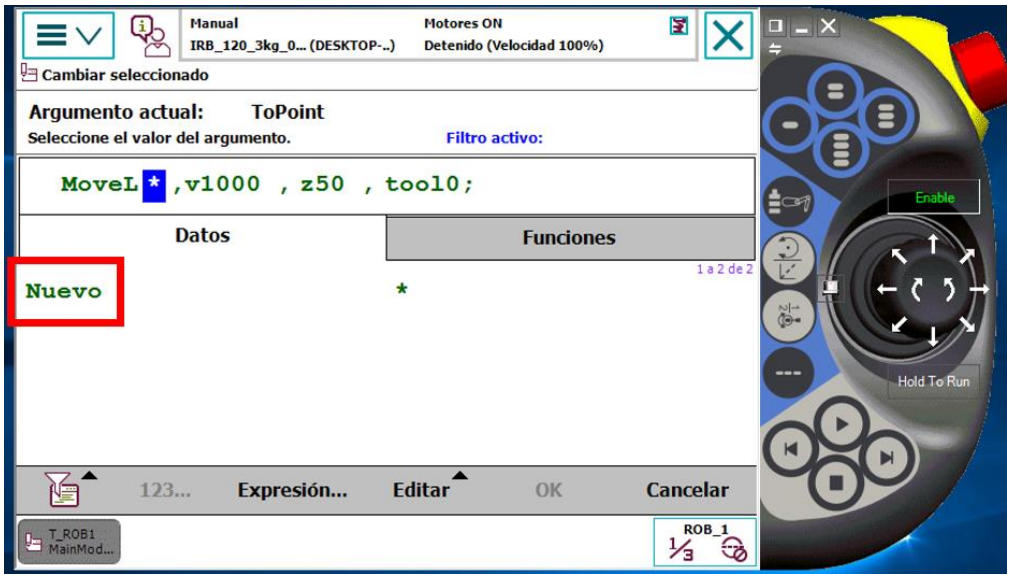


Figura 5-72. Creación de punto de posición.

Para crear un nuevo punto será necesario determinar los siguientes parámetros:

- Nombre de la posición
- Ámbito : si es local o global
- Tipo de almacenamiento :si es variable, persistente o constante
- Tarea : manipulador al que va asociado
- Modulo : modulo al que va asociado
- Valores (x,y,z): para definir estos valores será necesario seleccionar el botón “Valor incial”

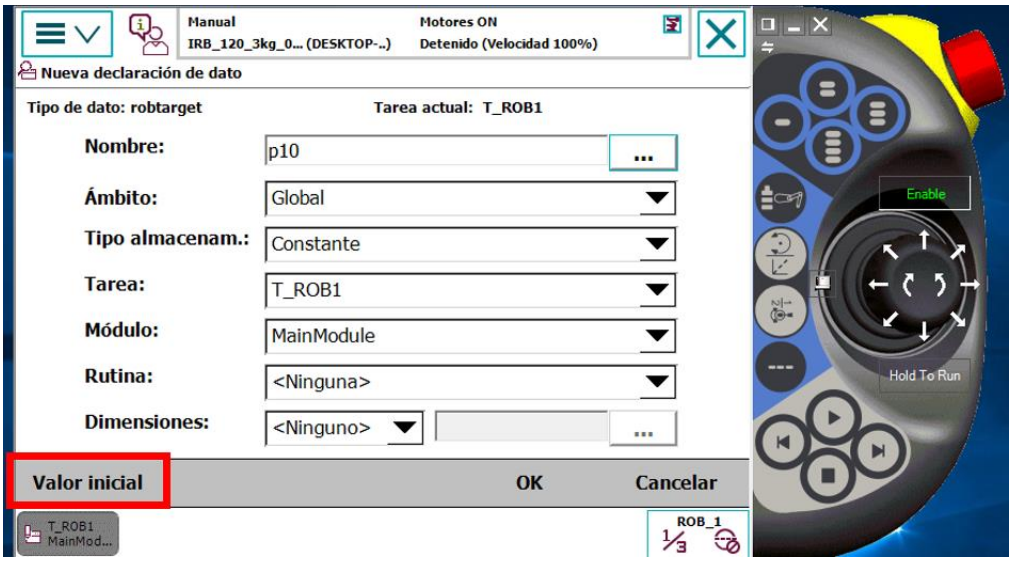


Figura 5-73. Definición de un nuevo punto.

Esta opción te permite introducir manualmente los valores de x, y, z además de los cuaternios.

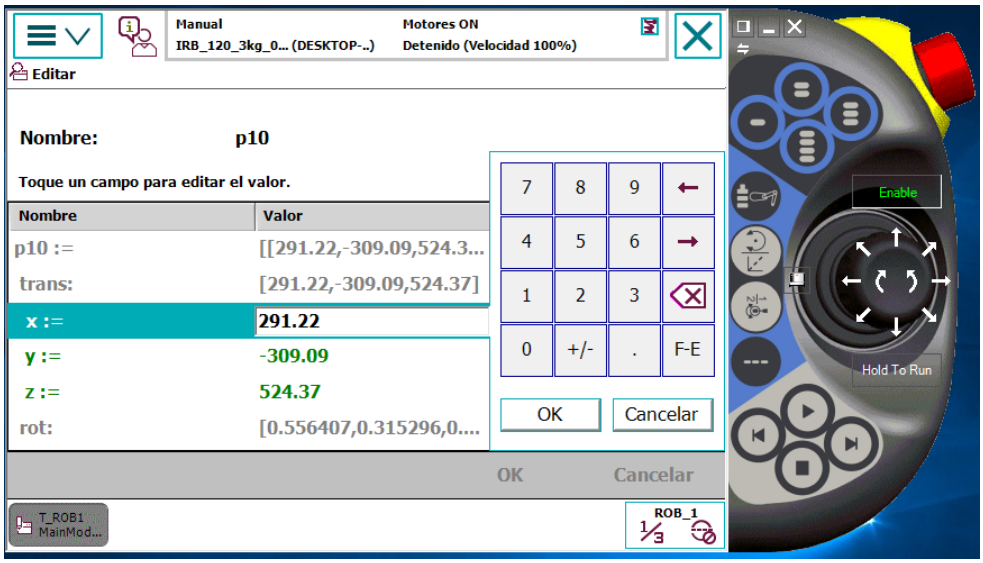


Figura 5-74. Introducción manual de las coordenadas x,y ,z del nuevo punto.

5.8.3.4 Datos del programa

Se pueden definir distintos datos que se van a utilizar en el programa (carga útil , herramientas, campo de trabajo)

Para definir una herramienta primero se selecciona la opción “tooldata”

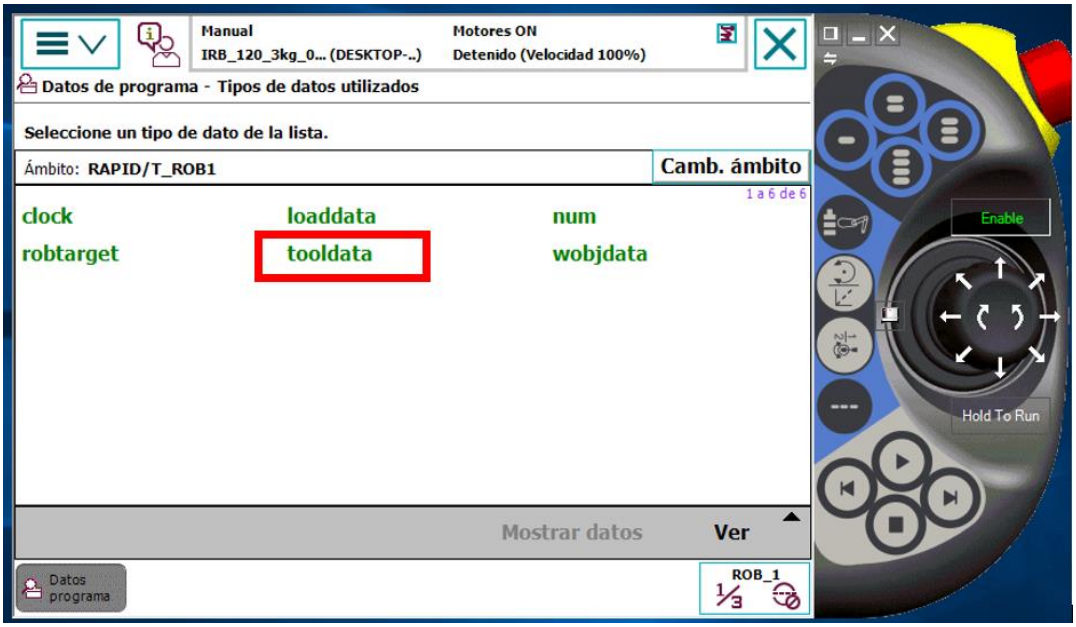


Figura 5-75. Libreria de herramientas definidas.

Una vez seleccionada la opción de la herramienta para crear una nueva será necesario seleccionar el botón “Nuevo”

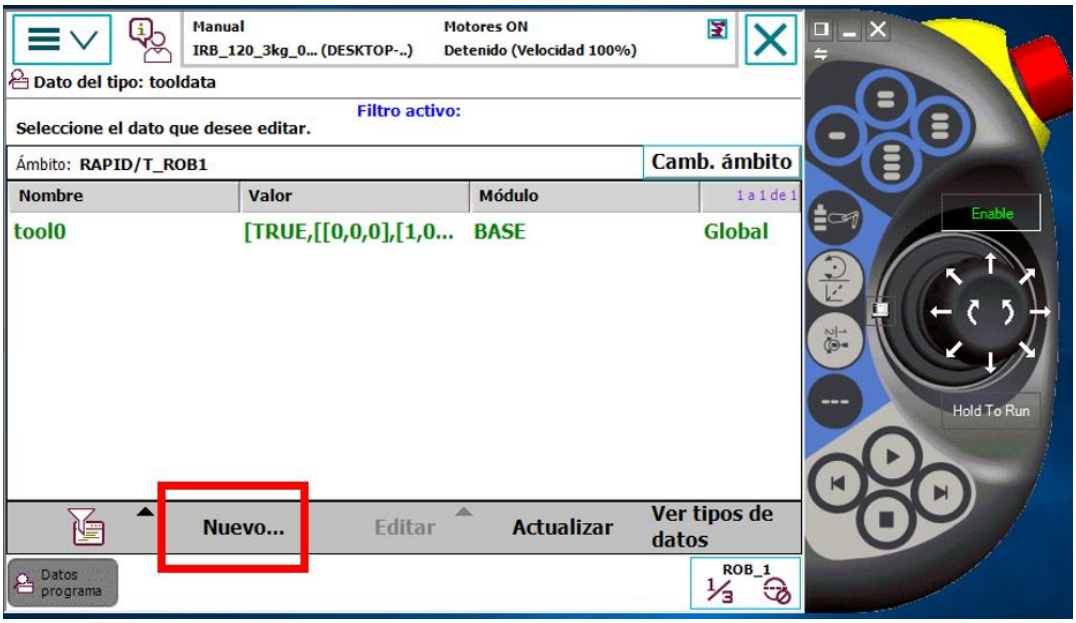


Figura 5-76. Creación de una nueva herramienta.

Los siguientes pasos para definir la herramienta son idénticos que la manera de definir una posición.

1. Definir la herramienta

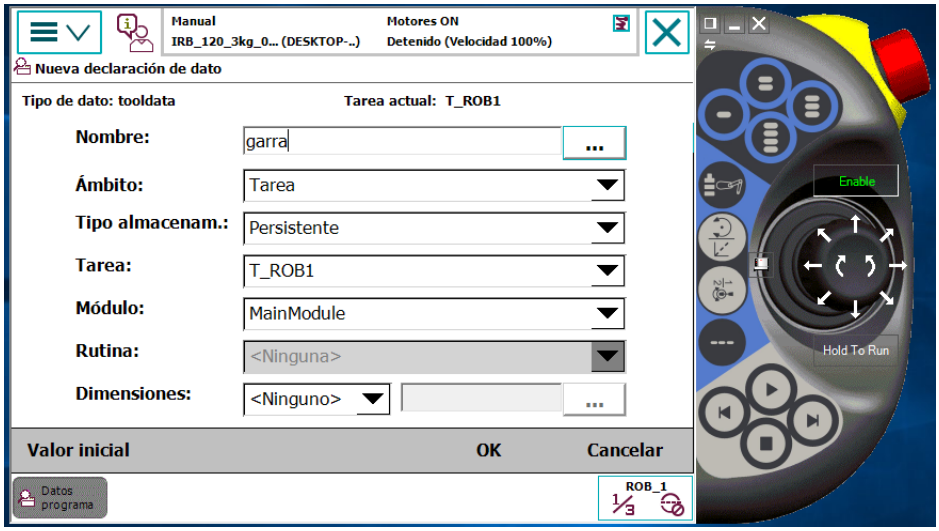


Figura 5-77. Definición de la herramienta.

2. Especificar las características de la herramienta

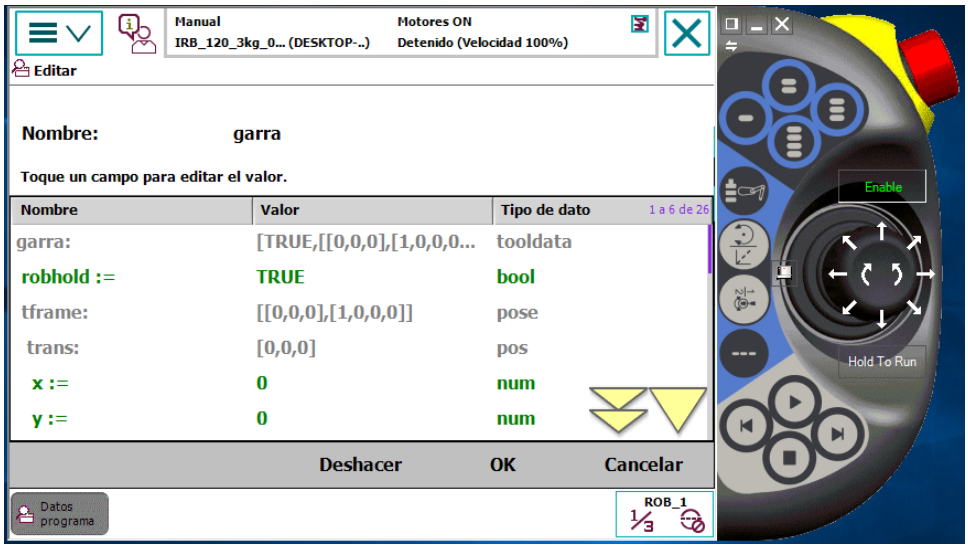


Figura 5-78. Especificación de las caracteristicas de la herramienta.

Una vez creada la herramienta aparecerá en el listado para poder ser usada posteriormente

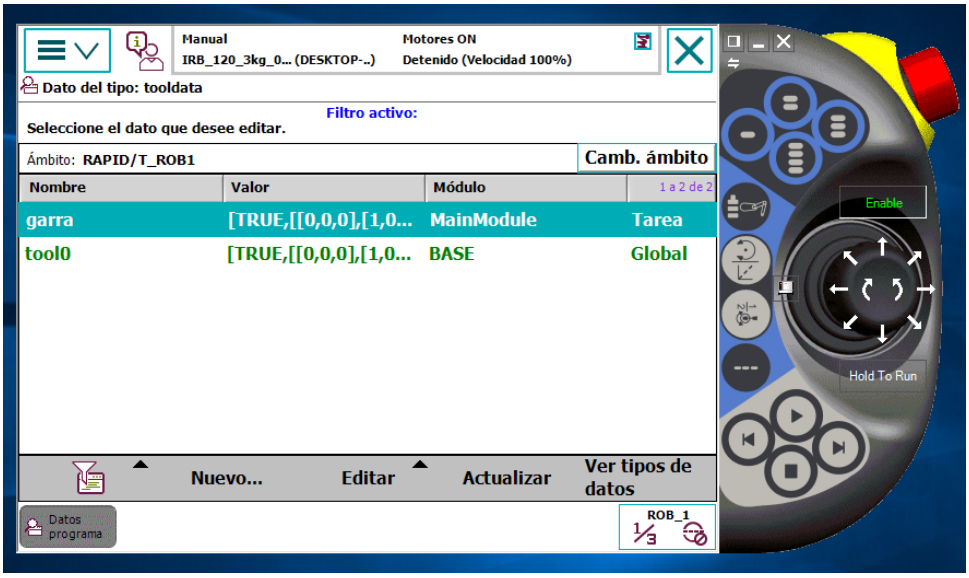


Figura 5-79. Listado de herramientas definidas.

Es importante uan vez definida la herramienta definir también la base de coordenadas del TCP de dicha herramienta seleccionando del método que más convenga en cada caso.

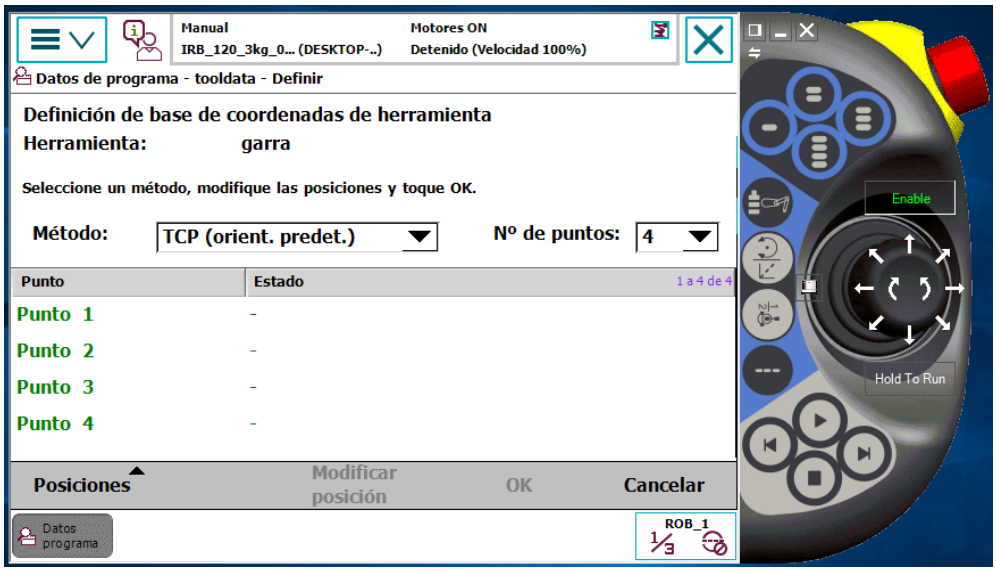


Figura 5-80. Definición de la base de coordenadas de la herramienta.

5.8.4 Configuración de las teclas programables del Flexpendant

El flexpendant cuenta con la opción de simular mediante las teclas programables salidas o entradas de sensores (cambio de estado de 0 a 1).

Para configurar dichas techas es necesario acceder al panel de control del menú principal del flexpendant y seleccionar la opción “teclas programables”

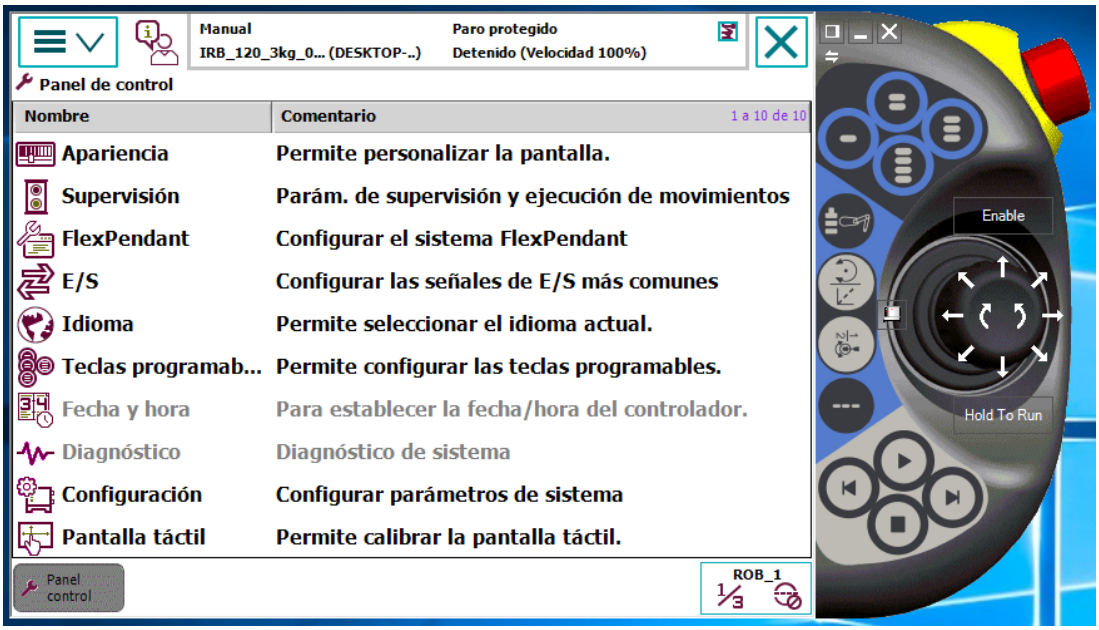


Figura 5-81. Opciones de configuración del panel de control.

Desde este menú podremos configurar los 4 botones independientemente

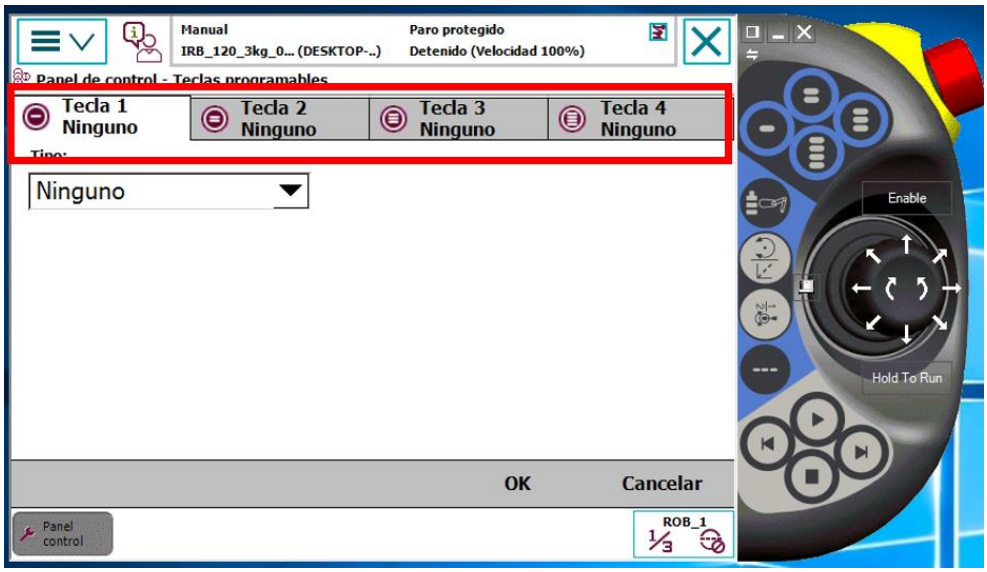


Figura 5-82. Configuración teclas programables.

Escogiendo el tipo de comportamiento que se desea para esa tecla (ninguno, entrada, salida, sistema).

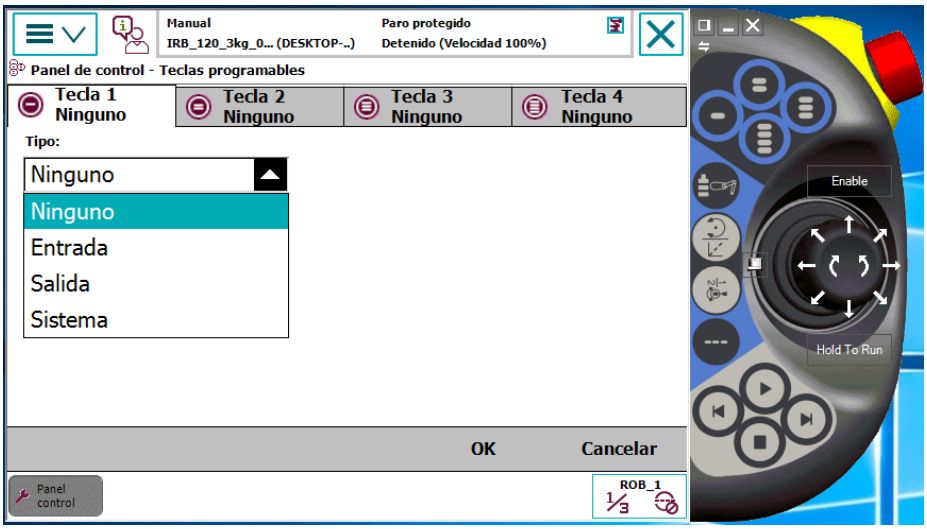


Figura 5-83. Tipos de función de las teclas programables.

Una vez seleccionado el tipo se determinará el comportamiento de la misma al ser presionada (Activar / desactivar, cambiar a 1, cambiar a 0, presionar/liberar, pulso)

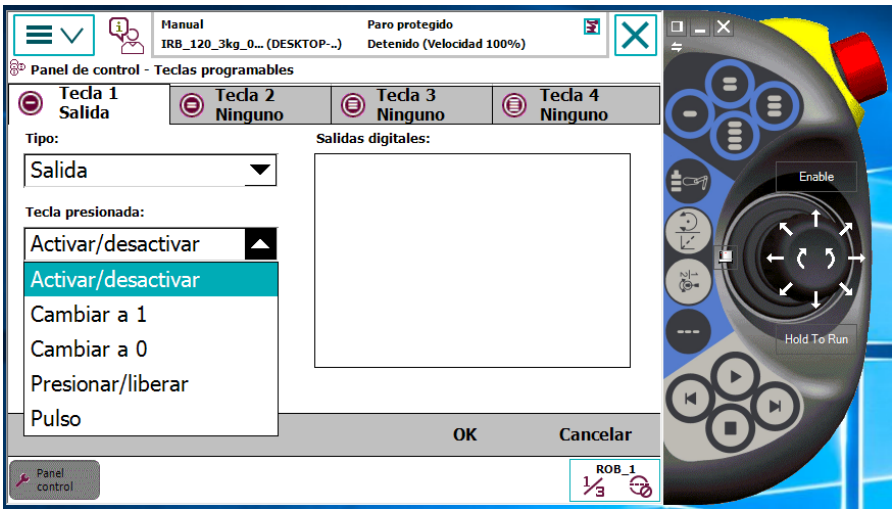


Figura 5-84. Tipos de comportamientos de las teclas programables.

Una vez configurado todo se pulsará ok para guardar los cambios.

5.8.5 Creación de módulos de E/S

Una vez creadas las geometrías de la estación, los controladores además de dotarlos de inteligencia, es necesario crear un módulo de entradas y salidas. Para esto, dentro de la Pestaña Controlador y en la ventana del navegador se despliega el apartado de *"Configuración"* y dentro de éste en *"I/O System"*. Se abrirá una ventana anexa con un listado de elementos a configurar dentro de dicho controlador.

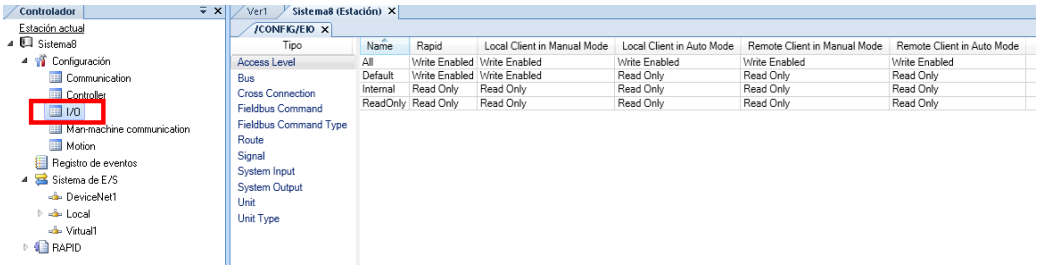


Figura 5-85. Conficuración de I/O System.

Para establecer las señales de entrada y salida asociadas a la unidad del controlador se crearán un conjunto de señales de entrada digitales que corresponderán a las señales de salida de los componentes inteligentes que sean necesarias para el control del robot.

Para ellos primero se deberá crear una unidad virtual de entradas y salidas y posteriormente crear las entradas y salidas asociadas a dicha undad virtual.

Para crear la unidad vitual se accederá al campo "Unit" y se seleccionará con el botón derecho "Nuevo unit"

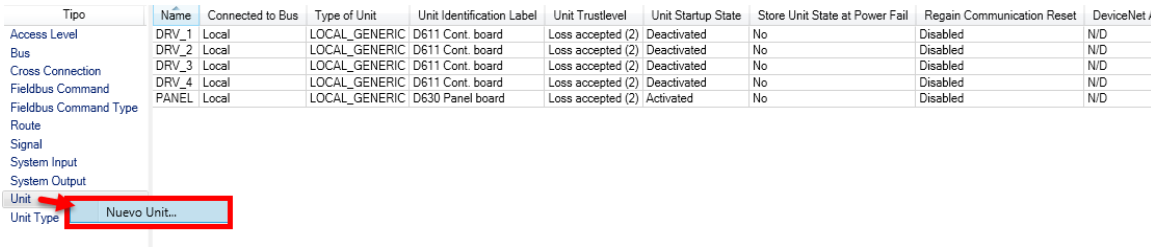


Figura 5-86. Creación nueva unidad.

Se configurará la unidad definiendo la comunicación con el bus correspondiente, el tipo de unidad y comportamiento que se desea en dicha unidad.

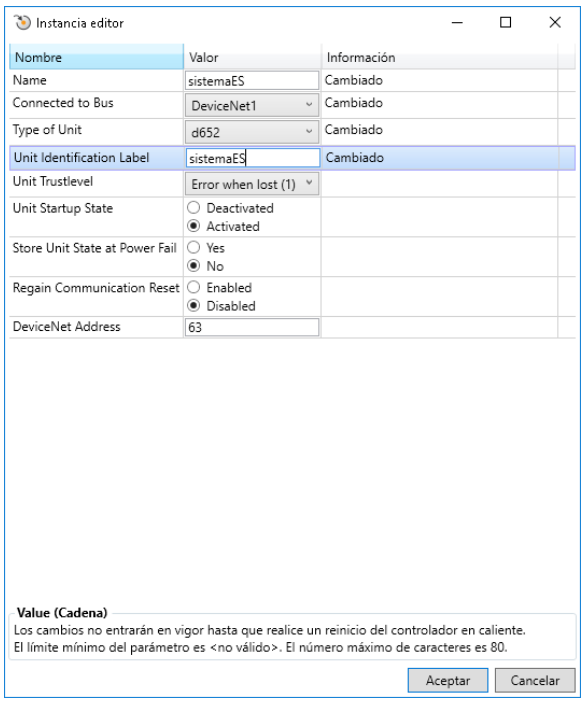


Figura 5-87. Configuración nueva unidad.

Para crear las entradas o salidas asociadas al controlador hay que seleccionar con el botón derecho el campo *"Signal"*, dentro de configuración e I/O, pulsando sobre *"Nuevo Signal"*.

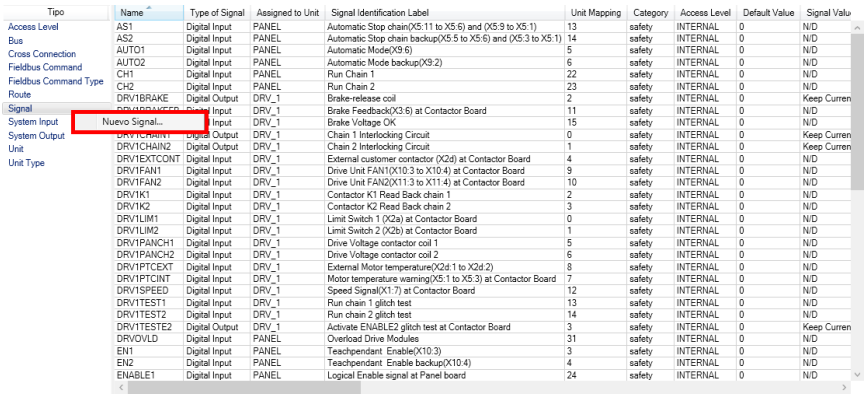


Figura 5-88.Creación de una nueva señal de entrada o salida.

Posteriormente se debe rellenar los datos requeridos para editar la señal correspondiente.

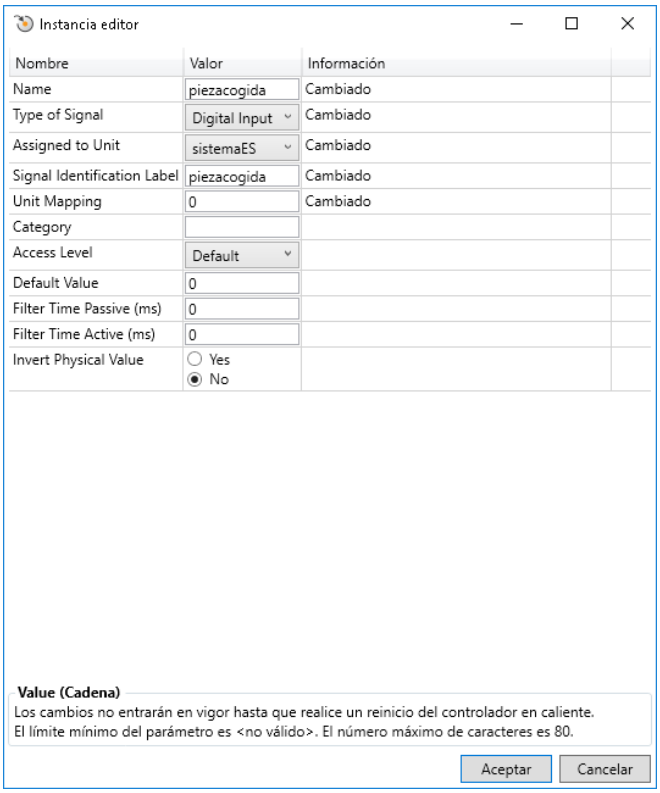


Figura 5-89.Introducción de una nueva señal de entrada o salida.

5.9 Menú Rapid

El menú Rapid se divide en 6 grupos diferentes que acogen diversas funciones:

- **Acceso :** sincronizar objetos de a estación con un código RAPID
- **Editar:** herramientas de edición del programa
- **Insertar:** para insertar fragmentos o instrucciones de programación
- **Buscar:** permite buscar y reemplazar de manera rápida algún comando de la programación.
- **Controlador:** aplicar cambios en todos los módulos modificados, enumerar las tareas activas en el controlador, seleccionar el modo de ejecución del controlador, opciones edición del programa, ajuste de robtargets.

- **Probar y depurar:** inicio de la ejecución de las tareas de RAPID en el sistema, tipos de ejecución de las sentencias, verificación del sistema

5.9.1 Creación de un módulo de programación

Para comenzar a programar la lista de instrucciones que defina la tarea del manipulador será necesario crear un nuevo módulo. Para ello, se accederá al menú Rapid y se seleccionará el icono de "*Programa*" desplegando el submenú con la flecha que aparece a la derecha del icono. A continuación se seleccionará "*Nuevo módulo*".

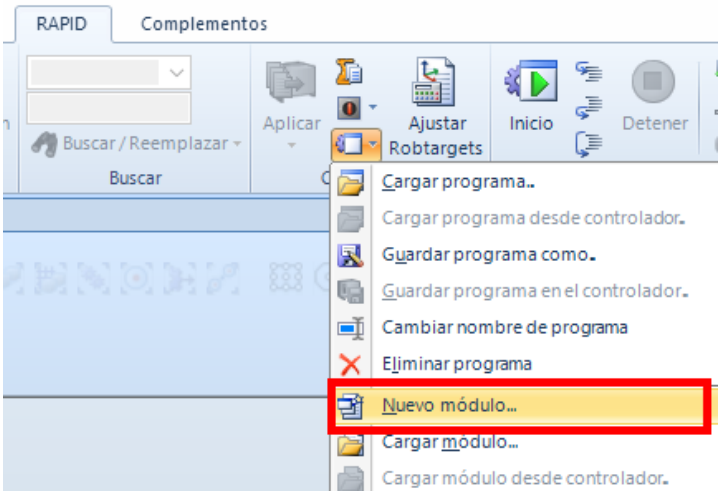


Figura 5-90.Introducción de un nuevo módulo.

Se definirá el nombre del módulo y el tipo de módulo; también se puede determinar los atributos del módulo.

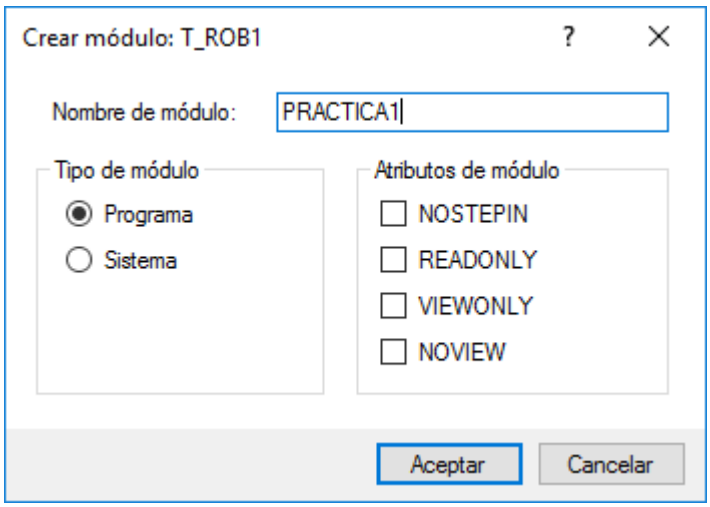


Figura 5-91.Definición del nuevo módulo.

Se accederá a dicho módulo a través de la ventana navegador, abriendo así una ventana nueva de edición del programa donde se trabajará programando las instrucciones necesarias para la tarea requerida.

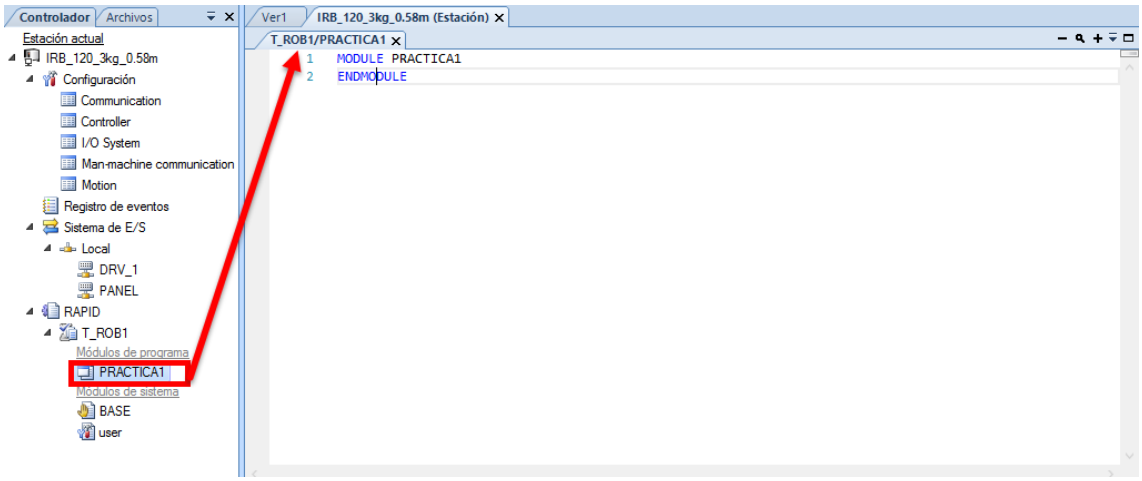


Figura 5-92.Ventana de programación del módulo creado.

5.9.2 Verificar programa

Una vez esté finalizado el código del programa será conveniente verificar dicho programa para depurar y comprobar que no existen errores que puedan provocar que no funcione. Para ello, se seleccionará la función *"Verificar programa"* dentro del menú RAPID.

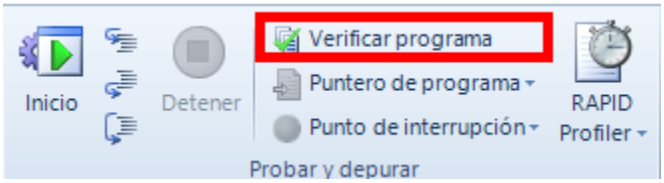


Figura 5-93.Verificación de programa.

5.9.3 Ejecución del programa

Si el programa ha sido depurado y ya no existen errores en el código que impidan la ejecución del mismo se procederá la simulación de la tarea seleccionando la función *"Inicio"* del menú RAPID.

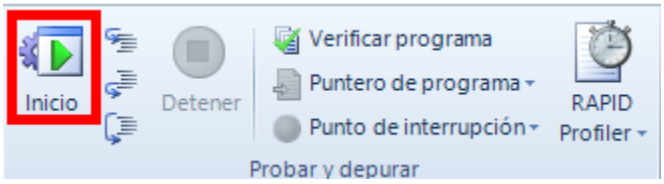


Figura 5-94.Inicio del código del programa.

6 EJERCICIOS PRACTICOS

EN este capitulo se realizarán una serie de ejercicios prácticas que se llevarán acabo en el laboratorio de la universidad empleando la simulación del programa RobotStudio.

6.1 Práctica 1: Mover una pieza desde una posición a otra

Coger una pieza de una posición porigen y llevarla a otra destino pdestino empleando posiciones de aproximación.

a) Hacerlo como una sucesión de comandos en modo directo desde el flexpendat. Definir las posiciones de aproximación manualmente.

b) Hacerlo en forma de programa. Definir las posiciones de aproximación relativas a porigen y pdestino.

OBJETIVO: conocer el sistema y los comandos básicos para el manejo manipulador, empleando el comando MoveL y MoveJ

RESOLUCION:

```
MODULE practica1
! definicion de herramienta garra
PERS tooldata wrist0:=[TRUE,[[0,0,129],[1,0,0,0]],[1,[0,0,100],[1,0,0,0],0,0,0]];
!Punto home base
CONST robtarget
home:=[[476.071080735,0,529.499955962],[0.49999989,0,0.866025467,0],[0,0,0,0],[9E9,9E
9,9E9,9E9,9E9,9E9]];
! Punto origen
CONST robtarget porig:=[[570,-130,260],[0.707106781,0,0.707106781,0],[-1,-
1,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
 ! Punto destino
CONST robtarget pdest:=[[570,150,260],[0.707106781,0,0.707106781,0],[0,0,-
1,1],[9E9,9E9,9E9,9E9,9E9,9E9]];

   PROC main()
        !Definición variables puntos
        VAR robtarget paori;
        VAR robtarget pades;
        MoveJ home,v800,fine,wrist0\WObj:=wobj0;
        paori:=Offs(porig,0,0,-20);
        pades:=Offs(pdest,0,0,-20);
        abrir;
        MoveJ paori,v800,fine,wrist0\WObj:=wobj0;
        MoveL porig,v400,fine,wrist0\WObj:=wobj0;
        cerrar;
        MoveL paori,v400,fine,wrist0\WObj:=wobj0;
        Movej pades,v600,fine,wrist0\WObj:=wobj0;
```

```
        MoveL pdest,v400,fine,wrist0\WObj:=wobj0;
        abrir;
        MoveL pades,v400,fine,wrist0\WObj:=wobj0;
        MoveJ home,v800,fine,wrist0\WObj:=wobj0;
        ENDPROC
        PROC abrir()
        waittime 5;
        reset doGripper;
        waittime 5;
        ENDPROC
        PROC cerrar()
        waittime 5;
        set doGripper;
        waittime 5;
        ENDPROC
ENDMODULE
```

6.2 Práctica 2: Seguimiento del contorno de una pieza

Supóngase una pieza cúbica orientada de acuerdo con los ejes X e Y del robot. En la variable lado vendrá dada la longitud del lado de la pieza. Se supone definida únicamente la posición de una de las esquinas de la pieza. Realizar un programa de manera que el extremo del robot:

a) Recorra las cuatro aristas superiores de dicha pieza en sentido horario.

b) Recorra en forma circular el contorno de la cara superior de la pieza, de nuevo en sentido horario.

OBJETIVO: Realizar trayectorias con los comandos MoveL y MoveC, empleando el comando TPReadFK y Test para la selección de operaciones.

RESOLUCION:

El mismo programa contendrá ambos apartados:

```
MODULE practica2

PERS tooldata
MyTool:=[TRUE,[[31.792631019,0,229.638935148],[0.945518576,0,0.325568154,0]],[1,[0,0
,1],[1,0,0,0],0,0,0]];
CONST robtarget
home:=[[547.330665099,0,451.64730631],[0.190808996,0,0.981627183,0],[0,-
1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
! definición punto Inicio Arista
        CONST robtarget p2:=[[400,0,200],[0,0,1,0],[0,0,-1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
! Definición puntos circulo
   CONST robtarget p10:=[[600,100,200],[0,0,1,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
   CONST robtarget p20:=[[500,200,200],[0,0,1,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
   CONST robtarget p30:=[[400,100,200],[0,0,1,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
   CONST robtarget p40:=[[500,0,200],[0,0,1,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
!Variable zone para el punto de paso del circulo
```

```
VAR ZONEDATA zcirculo:=[false,1,0,0,0,0,0];
PROC main()
        VAR num operacion;
        VAR num lado;
        VAR num n_r;

!Mover a home
        MoveJ home,v800,fine,MyTool\WObj:=wobj0;
!Limpiar pantalla
        TPErase;
       TPReadFK operacion,"Seleccione operacion a
realizar","Aristas","Circulo","","","Salir";
        TEST operacion
            CASE 1:
                TPReadNum lado,"Introducir lado del cubo ";
                TPReadNum n_r,"Introducir numero veces a ejecutar";
                Aristas lado,n_r;
            CASE 2:
                TPReadNum n_r,"Introducir numero veces a ejecutar";
                Circulo n_r;
           CASE 5:
                STOP;
                DEFAULT:
                TPWrite "Elección ilegal";
        ENDTEST
ENDPROC
PROC Aristas(VAR num lado1,VAR num veces)
!Definición variables puntos

        VAR robtarget pb;
        VAR robtarget pc;
        VAR robtarget pd;
!Puntos cuadrado

        pb:=Offs(p2,lado1,0,0);
        pc:=Offs(pb,0,lado1,0);
        pd:=Offs(pc,-lado1,0,0);
!Bucle repeticiones sin volver en cada una a pHome
        FOR i FROM 1 TO veces DO
!Movimientos cuadrado
            MoveJ p2,v800,fine,MyTool\WObj:=wobj0;
            MoveL pb,v800,fine,MyTool\WObj:=wobj0;
            MoveL pc,v800,fine,MyTool\WObj:=wobj0;
            MoveL pd,v800,fine,MyTool\WObj:=wobj0;
            MoveL p2,v800,fine,MyTool\WObj:=wobj0;
        ENDFOR
        MoveJ home,v800,fine,MyTool\WObj:=wobj0;
ENDPROC
PROC CIRCULO(VAR num veces)
        FOR i FROM 1 TO veces DO
!Movimientos circulo
```

```
        MoveJ p10,v800,fine,MyTool\WObj:=wobj0;
        MoveC p20,p30,v800,zcirculo,MyTool\WObj:=wobj0;
        MoveC p40,p10,v800,fine,MyTool\WObj:=wobj0;
    ENDFOR
    MoveJ home,v800,fine,MyTool\WObj:=wobj0;
ENDPROC

ENDMODULE
```

6.3 Práctica 3: Formar una torre de piezas

Para formar la torre se tomarán medidas de las alturas de las piezas.

a) Programa que traslade una torre de tres piezas de un lugar a otro, debiendo quedar las piezas de la torre destino en la misma posición relativa que estaban en la torre original. Para ello, las piezas serán recogidas de una torre vertical previamente formada, de la cual se enseñará al robot solo la posición de la base. La posición de la torre destino deberá ser 15 cm a la derecha de la posición de la torre inicial.

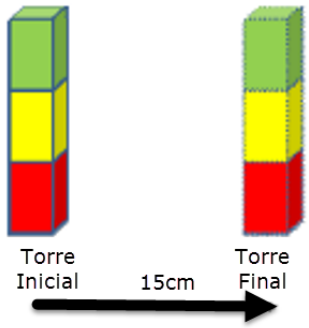


Figura 6-1.Práctica 3 a).

OBJETIVO: manejo del bucle For y comunación con el Flexpendat para introducción de variables usando comando TPReadnum

RESOLUCION:

```
MODULE practica3a
! definicion de herramienta garra
PERS tooldata wrist0:=[TRUE,[[0,0,129],[1,0,0,0]],[1,[0,0,100],[1,0,0,0],0,0,0]];
!Punto home base
CONST robtarget
home:=[[476.071080735,0,529.499955962],[0.49999989,0,0.866025467,0],[0,0,0,0],[9E9,9E
9,9E9,9E9,9E9,9E9]];
! Punto origen
CONST robtarget p50:=[[570,-130,260],[0.707106781,0,0.707106781,0],[-1,-
1,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
 ! Punto destino
CONST robtarget p70:=[[570,150,260],[0.707106781,0,0.707106781,0],[0,0,-
1,1],[9E9,9E9,9E9,9E9,9E9,9E9]];

PROC main()
```

```
!Definición variables puntos
VAR robtarget P51
VAR robtarget P52
VAR robtarget P53
VAR robtarget P61
VAR robtarget P62
VAR robtarget P63
VAR robtarget P71
VAR robtarget P72
VAR robtarget P73
!offset de los puntos empleados
p51:=Offs(p50,0,0,0);
p52:=Offs(p50,0,0,40);
p53:=Offs(p52,0,0,40);
p61:=Offs(p50,0,100,0);
p62:=Offs(p50,0,100,40);
p63:=Offs(p62,0,100,40);
p71:=Offs(p70,0,0,0);
p72:=Offs(p70,0,0,40);
p73:=Offs(72,0,0,40);

!Mover a home
MoveJ home,v400,fine, wrist0\WObj:=wobj0;
abrir;

!Bucle repeticiones sin volver en cada una a pHome
   MoveL p53,v400,fine, wrist0\WObj:=wobj0;
   cerrar;
   MoveL p71,v400,fine, wrist0\WObj:=wobj0;
   abrir;
   MoveL p52,v400,fine, wrist0\WObj:=wobj0;
   cerrar;
   MoveL p72,v400,fine, wrist0\WObj:=wobj0;
   abrir;
   MoveL p51,v400,fine, wrist0\WObj:=wobj0;
   cerrar;
   MoveL p73,v400,fine, wrist0\WObj:=wobj0;
   Waittime 5;
   MoveL p61,v400,fine, wrist0\WObj:=wobj0;
   abrir:
   MoveL p72,v400,fine, wrist0\WObj:=wobj0;
  cerrar;
   MoveL p62,v400,fine, wrist0\WObj:=wobj0;
   abrir;
   MoveL p71,v400,fine, wrist0\WObj:=wobj0;
   cerrar;
   MoveL p63,v400,fine, wrist0\WObj:=wobj0;
   abrir
   MoveL home,v800,fine, wrist0\WObj:=wobj0;
ENDPROC
PROC abrir()
        waittime 5;
        reset doGripper;
        waittime 5;
```

```
ENDPROC
PROC cerrar()
waittime 5;
set doGripper;
waittime 5;
ENDPROC
```

```
ENDMODULE
```

b) Programa que forme una torre de N piezas (N definida por teclado). Las piezas serán recogidas de unas posiciones en forma de estructura matricial de dos dimensiones, con una separación de 10 cm entre centros.

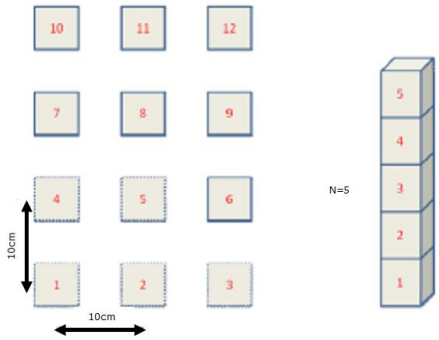


Figura 6-2.Práctica 3 b).

RESOLUCION:

```
MODULE practica5

! definicion de herramienta garra
PERS tooldata wrist0:=[TRUE,[[0,0,129],[1,0,0,0]],[1,[0,0,100],[1,0,0,0],0,0,0]];
!Punto home base
CONST robtarget
home:=[[476.071080735,0,529.499955962],[0.49999989,0,0.866025467,0],[0,0,0,0],[9E9,9E
9,9E9,9E9,9E9,9E9]];
! Punto origen
CONST robtarget P1:=[[570,-130,260],[0.707106781,0,0.707106781,0],[-1,-
1,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
! Punto almacenar
CONST robtarget P2:=[[570,150,260],[0.707106781,0,0.707106781,0],[0,0,-
1,1],[9E9,9E9,9E9,9E9,9E9,9E9]];

   VAR num off_z:= 30;
   VAR num off_c;
   VAR num off_f;
   VAR num cont_c;
   VAR num cont_f;
   VAR num i;
   VAR num j;
```

```
PROC main()
    MoveL home v800,fine, wrist0\WObj:=wobj0;
    abrir;
    TPErase;
    TPReadNum i,"Introducir posicion en de la columna (de 1 a 3) ";
    TPReadNum j,"Introducir posicion en de la fila (de 1 a 4) ";
    Bucle i,j;
    MoveL home v800,fine, wrist0\WObj:=wobj0;
ENDPROC
PROC Bucle(VAR num i, VAR num j)
    FOR cont_c FROM 0 TO i DO
        FOR cont_f FROM 0 TO j DO
            MoveL Offs(P1,off_c,off_f,off_z),v800,fine, wrist0\WObj:=wobj0;
            cerrar;
            MoveL Offs(P1,off_c,off_f,0),v800,fine, wrist0\WObj:=wobj0;
            MoveL Offs(P2,off_c,off_f,off_z*j) v800,fine, wrist0\WObj:=wobj0;
            abrir;
            WaitTime 1;
            off_f:= off_f+10;
        ENDFOR
        off_c:=off_c+10;
        off_f:=0;
    ENDFOR
    ENDPROC
PROC abrir()
        waittime 1;
        reset doGripper;
        waittime 1;
ENDPROC
PROC cerrar()
        waittime 1;
        set doGripper;
        waittime 1;
ENDPROC

ENDMODULE
```

6.4 Práctica 4: Seguimiento de trayectoria arbitraria. Ejemplo de aplicación de soldadura, sellado, corte, pegado….

Supóngase que se desea depositar una tira de pegamento sobre una trayectoria de forma senoidal, que comienza en la posición CENTR y cuyas amplitudes AMPX y AMPY vienen definidas como satos dentro del programa (véase en la figura adjunta)

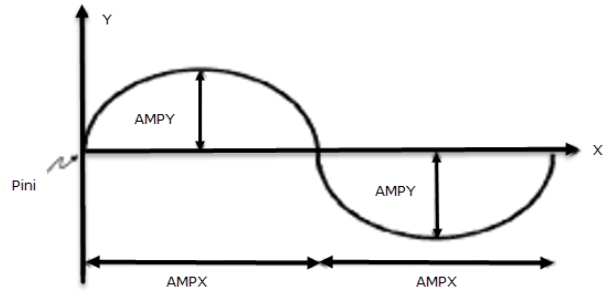


Figura 6-3.Práctica 4.

Realizar un programa que recorra un programa que recorra la trayectoria dos veces, una en sentido directo y otra en inverso.

OBJETIVO: realizar llamada a procedimientos introduciendo variables atraves del flexpendat para realizar trayectorias empleando Offset para la definición de posiciones.

RESOLUCION:

```
MODULE practica4
   PERS tooldata
MyTool:=[TRUE,[[31.792631019,0,229.638935148],[0.945518576,0,0.325568154,0]],[1,[0,0
,1],[1,0,0,0],0,0,0]];
        CONST robtarget
home:=[[547.331,0,451.647],[0.190809,0,0.981627,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,
9E+09,9E+09]];
        CONST robtarget CENTR:=[[603,-300,200],[0,0,1,0],[-1,0,-
1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
PROC main()
    VAR num AMPLIY;
    VAR num AMPLIX;
     VAR num n_r;

!Mover a home
       MoveJ home,v800,fine,MyTool\WObj:=wobj0;
!Limpiar pantalla
       TPErase;
       TPReadNum AMPLIY,"Introducir amplitud en el ejem Y de la trayectoria senoidal ";
       TPReadNum AMPLIX,"Introducir amplitud en el ejem X de la trayectoria senoidal";
       TPReadNum n_r,"Introducir numero veces a ejecutar la trayectoria de la soldadura";
            SENOIDE AMPLIY,AMPLIX,n_r;
ENDPROC
PROC SENOIDE (VAR num AMPX,VAR num AMPY,VAR num veces)
!Definición variables puntos

       VAR robtarget p1;
       VAR robtarget p2;
       VAR robtarget p3;
       VAR robtarget p4;
!Puntos para la senoide
```

```
        p1:=Offs(CENTR,AMPX,AMPY,0);
        p2:=Offs(p1,-AMPX,AMPY,0);
        p3:=Offs(p2,-AMPX,AMPY,0);
        p4:=Offs(p3,AMPX,AMPY,0);
!Bucle repeticiones sin volver en cada una a pHome
        FOR i FROM 1 TO veces DO

!Movimientos senoide
            MoveJ home,v800,fine,MyTool\WObj:=wobj0;
            MoveL CENTR,v300,fine,MyTool\WObj:=wobj0;
            MoveC p1,p2,v400,fine,MyTool\WObj:=wobj0;
            MoveC p3,p4,v400,fine,MyTool\WObj:=wobj0;
            WaitTime 2;
            MoveC p3,p2,v400,fine,MyTool\WObj:=wobj0;
            MoveC p1,CENTR,v400,fine,MyTool\WObj:=wobj0;
        ENDFOR
        MoveJ home,v800,fine,MyTool\WObj:=wobj0;
ENDPROC

ENDMODULE
```

6.5 Práctica 5: Uso de entradas digitales

Realizar un programa que coja una pieza en forma de bloque situada sobre una mesa y la coloque en una entre 4 posibles posiciones de destino. La selección de la posición final la hará el usuario mediante un código binario de dos dígitos. Solo se enseñará al robot la posición inicial mediante guiado manual, debiéndose calcular las posiciones de destino a partir de esta inicial. Las posiciones de destino y el código asociadas a cada una de ellas aparecen en la figura adjunta. Nótese que en las posiciones de destino, la pieza ha sido girada 180° respecto a su eje vertical. Los dígitos binarios serán leídos desde teclado.

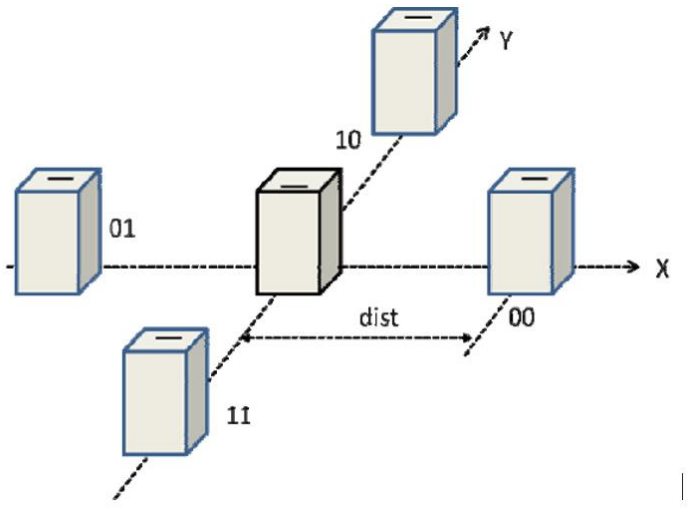


Figura 6-4.Práctica 5.

OBJETIVO: utilización de señales de entrada y salida para la ejecución de la aplicación, esmpleando la función Reltool para la definición de las posiciones junto con el giro de la pieza .

RESOLUCION:

```
MODULE Practica5a
    ! definicion de herramienta garra
    PERS tooldata wrist0:=[TRUE,[[0,0,129],[1,0,0,0]],[1,[0,0,100],[1,0,0,0],0,0,0]];
    !Punto home base
    CONST robtarget
    home:=[[476.071080735,0,529.499955962],[0.49999989,0,0.866025467,0],[0,0,0,0],[
    9E9,9E9,9E9,9E9,9E9,9E9]];
    ! Punto origen
    CONST robtarget centro:=[[570,-130,260],[0.707106781,0,0.707106781,0],[-1,-
    1,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    PROC main()
     VAR num posicion;
     VAR num dist;
    !Mover a home
     MoveJ phome,v800,fine, wrist0\WObj:=wobj0;
    !Limpiar pantalla
     TPErase;
     TPReadFK posicion,"Seleccione operacion a realizar","00","01","10","11","Salir";
     TEST operacion
          CASE 1:
             TPReadNum dist,"Introducir distacia entre la posición inicial y la de destino";
             P00 dist;
          CASE 2:
             TPReadNum dist,"Introducir distacia entre la posición inicial y la de destino";
             P01 dist;
         CASE 3:
             TPReadNum dist,"Introducir distacia entre la posición inicial y la de destino";
             P10 dist;
        CASE 4:
             TPReadNum dist,"Introducir distacia entre la posición inicial y la de destino";
             P11 dist;
         CASE 5:
             STOP;
             DEFAULT:
             TPWrite "Elección ilegal";
       ENDTEST
ENDPROC
PROC P00 (VAR num distancia)
!Definición variables puntos
     VAR robtarget pInicio;
     VAR robtarget pos00;
     VAR robtarget Pdef00;
!definicion de puntos
    pInicio:=Offs(CENTRO,0,0,-100);
    pos00:=RelTool(pInicio,distancia,0,0,\Rz:=180);
    Pdef00:=Offs(pos00,0,0,-100);
!Movimientos
```

```
        MoveJ CENTRO,v800,fine, wrist0\WObj:=wobj0;
        MoveL pInicio,v300,fine, wrist0\WObj:=wobj0;
        MoveL CENTRO,v300,fine, wrist0\WObj:=wobj0;
        MoveL pos00,v800,fine, wrist0\WObj:=wobj0;
        MoveL Pdef00,v300,fine, wrist0\WObj:=wobj0;
        MoveL pos00,v800,fine, wrist0\WObj:=wobj0;
        MoveJ phome,v800,fine, wrist0\WObj:=wobj0;
    ENDPROC
PROC P01 (VAR num distancia)
!Definición variables puntos
        VAR robtarget pInicio;
        VAR robtarget pos01;
        VAR robtarget Pdef01;
!definicion de puntos
        pInicio:=Offs(CENTRO,0,0,-100);
        pos01:=RelTool(pInicio,-distancia,0,0,\Rz:=180);
        Pdef01:=Offs(pos01,0,0,-100);
!Movimientos
        MoveJ CENTRO,v800,fine, wrist0\WObj:=wobj0;
        MoveL pInicio,v300,fine, wrist0\WObj:=wobj0;
        MoveL CENTRO,v300,fine, wrist0\WObj:=wobj0;
        MoveL pos01,v800,fine, wrist0\WObj:=wobj0;
        MoveL Pdef01,v300,fine, wrist0\WObj:=wobj0;
        MoveL pos01,v800,fine, wrist0\WObj:=wobj0;
        MoveJ phome,v800,fine, wrist0\WObj:=wobj0;
ENDPROC
PROC P10 (VAR num distancia)
!Definición variables puntos
        VAR robtarget pInicio;
        VAR robtarget pos10;
        VAR robtarget Pdef10;
!definicion de puntos
        pInicio:=Offs(CENTRO,0,0,-100);
        pos10:=RelTool(pInicio,0,distancia,0,\Rz:=180);
        Pdef10:=Offs(pos10,0,0,-100);
!Movimientos
        MoveJ CENTRO,v800,fine, wrist0\WObj:=wobj0;
        MoveL pInicio,v300,fine, wrist0\WObj:=wobj0;
        MoveL CENTRO,v300,fine, wrist0\WObj:=wobj0;
        MoveL pos10,v800,fine, wrist0\WObj:=wobj0;
        MoveL Pdef10,v300,fine, wrist0\WObj:=wobj0;
        MoveL pos10,v800,fine, wrist0\WObj:=wobj0;
        MoveJ phome,v800,fine, wrist0\WObj:=wobj0;
ENDPROC
PROC P11 (VAR num distancia)
!Definición variables puntos
        VAR robtarget pInicio;
        VAR robtarget pos11;
        VAR robtarget Pdef11;
!definicion de puntos
        pInicio:=Offs(CENTRO,0,0,-100);
        pos11:=RelTool(pInicio,0,-distancia,0,\Rz:=180);
        Pdef11:=Offs(pos11,0,0,-100);
```

```
!Movimientos
        MoveJ CENTRO,v800,fine, wrist0\WObj:=wobj0;
        MoveL pInicio,v300,fine, wrist0\WObj:=wobj0;
        MoveL CENTRO,v300,fine, wrist0 \WObj:=wobj0;
        MoveL pos11,v800,fine, wrist0\WObj:=wobj0;
        MoveL Pdef11,v300,fine, wrist0WObj:=wobj0;
        MoveL pos11,v800,fine, wrist0\WObj:=wobj0;
        MoveJ phome,v800,fine, wrist0\WObj:=wobj0;
ENDPROC

ENDMODULE
```

6.6 Práctica 6: gestion de almacén de piezas

Realizar un programa que sea capaz de gestionar un almacén de piezas. A modo de ejemplo, considérese el sistema robótico utilizado en algunas farmacias.

Se considerarán ocho compartimentos, cuyas posiciones serán calculadas a partir de una posición inicial única enseñada mediante guiado manual, donde apilar verticalmente distintos tipos de piezas. Se considerará que todas las piezas apiladas en un mismo compartimento serán iguales.

Se generarán códigos de órdenes de manera que el usuario pueda elegir entre almacenar o retirar una pieza de un tipo concreto, utilizando como interfaz la pantalla y el teclado para mostrar e introducir la información necesaria.

Adicionalmente se utilizará una entrada digital a modo de interruptor de seguridad (por ejemplo, para el caso de que un usuario se introduzca en el espacio de trabajo). Así, en caso de que se active esta entrada, los programas se detendrán hasta que esta entrada vuelva a ser desactivada.

OBJETIVO: utilización de señales de entrada y salida para la ejecución de la aplicación y comunicación del robot con la herramienta garra; utilizando el flexpendat para la selección de la ubicación.

RESOLUCION:

```
MODULE Practica8
! definicion de herramienta garra
PERS tooldata wrist0:=[TRUE,[[0,0,129],[1,0,0,0]],[1,[0,0,100],[1,0,0,0],0,0,0]];
!Punto home base
CONST robtarget
home:=[[476.071080735,0,529.499955962],[0.49999989,0,0.866025467,0],[0,0,0,0],[9E9,9E
9,9E9,9E9,9E9,9E9]];
! Punto origen
CONST robtarget P1:=[[570,-130,260],[0.707106781,0,0.707106781,0],[-1,-
1,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
 ! Punto almacenar
CONST robtarget palmacen:=[[570,150,260],[0.707106781,0,0.707106781,0],[0,0,-
1,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
! Punto retirar
CONST robtarget pretirar:=[[570,230,260],[0.707106781,0,0.707106781,0],[0,-
2,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
PROC main()
```

```
        VAR num operacion;
        VAR num Tipo;
        VAR num Tipo1;

!Mover a home
        MoveJ home,v800,fine, wrist0\WObj:=wobj0;
        abrir;
!Limpiar pantalla
        TPErase;
        TPReadFK operacion,"Seleccione operacion a realizar","Almacenar","Retirar","","","Salir";
        TEST operacion
            CASE 1:
                TPReadFK Tipo,"Seleccione el tipo de producto ","Tipo 1","Tipo 2","Tipo 3","Tipo
4","Otros tipos";
             TEST operacion
                        CASE 1:
                        MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                        cerrar;
                        MoveL P1,v800,fine, wrist0\WObj:=wobj0;
                        abrir;
                        MoveL home,v800,fine, wrist0 \WObj:=wobj0;
                        CASE 2:
                        MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                        cerrar;
                        MoveL Offs(P1,20,0,0),v800,fine, wrist0\WObj:=wobj0;
                        abrir;
                        MoveL home,v800,fine, wrist0\WObj:=wobj0;
                CASE 3:
                        MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                        cerrar;
                        MoveL Offs(P1,40,0,0),v800,fine, wrist0\WObj:=wobj0;
                        abrir;
                        MoveL home,v800,fine, wrist0\WObj:=wobj0;
                CASE 4:
                        MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                        cerrar;
                        MoveL Offs(P1,60,0,0),v800,fine, wrist0\WObj:=wobj0;
                        abrir;
                        MoveL home,v800,fine, wrist0\WObj:=wobj0;
                CASE 5:
                         TPReadFK Tipo1,"Seleccione el tipo de producto ","Tipo 5","Tipo 6","Tipo
7","Tipo 8","";
                    TEST operacion
                        CASE 1:
                            MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                            cerrar;
                            MoveL Offs(P1,80,0,0),v800,fine, wrist0\WObj:=wobj0;
                            abrir;
                            MoveL home,v800,fine, wrist0\WObj:=wobj0;
                        CASE 2:
                            MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                            cerrar;
                            MoveL Offs(P1,100,0,0),v800,fine, wrist0\WObj:=wobj0;
```

```
                    abrir;
                    MoveL home,v800,fine, wrist0\WObj:=wobj0;
                CASE 3:
                    MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                    cerrar;
                    MoveL Offs(P1,120,0,0),v800,fine, wrist0\WObj:=wobj0;
                    abrir;
                    MoveL home,v800,fine, wrist0\WObj:=wobj0;
                CASE 4:
                    MoveJ Palmacen,v800,fine, wrist0\WObj:=wobj0;
                    cerrar;
                    MoveL Offs(P1,140,0,0),v800,fine, wrist0\WObj:=wobj0;
                    abrir;
                    MoveL home,v800,fine, wrist0\WObj:=wobj0;
                ENDTEST
        ENDTEST
         CASE 2:
            TPReadFK Tipo,"Seleccione el tipo de producto que desee","Tipo 1","Tipo 2","Tipo
3","Tipo 4","Otros tipos";
          TEST operacion
              CASE 1:
                  MoveL P1,v800,fine, wrist0\WObj:=wobj0;
                  cerrar;
                  MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                  abrir;
                  MoveL home,v800,fine, wrist0\WObj:=wobj0;
              CASE 2:
                  MoveL Offs(P1,20,0,0),v800,fine, wrist0\WObj:=wobj0;
                  cerrar;
                  MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                  abrir;
                  MoveL home,v800,fine, wrist0\WObj:=wobj0;
          CASE 3:
                  MoveL Offs(P1,40,0,0),v800,fine, wrist0\WObj:=wobj0;
                  cerrar;
                  MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                  abrir;
                  MoveL home,v800,fine, wrist0\WObj:=wobj0;
                  CASE 4:
                  MoveL Offs(P1,60,0,0),v800,fine, wrist0\WObj:=wobj0;
                  cerrar;
                  MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                  abrir;
                  MoveL home,v800,fine, wrist0WObj:=wobj0;
              CASE 5:
                  TPReadFK Tipo1,"Seleccione el tipo de producto que desee","Tipo 5","Tipo
6","Tipo 7","Tipo 8","";
            TEST operacion
               CASE 1:
                    MoveL Offs(P1,80,0,0),v800,fine, wrist0\WObj:=wobj0;
                    cerrar;
                    MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                    abrir;
                    MoveL home,v800,fine, wrist0\WObj:=wobj0;
```

```
                CASE 2:
                        MoveL Offs(P1,100,0,0),v800,fine, wrist0\WObj:=wobj0;
                        cerrar;
                        MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                        abrir;
                        MoveL home,v800,fine, wrist0\WObj:=wobj0;
                CASE 3:
                        MoveL Offs(P1,120,0,0),v800,fine, wrist0\WObj:=wobj0;
                        cerrar;
                        MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                        abrir;
                        MoveL home,v800,fine, wrist0\WObj:=wobj0;

                CASE 4:
                        MoveL Offs(P1,140,0,0),v800,fine, wrist0\WObj:=wobj0;
                        cerrar;
                        MoveL pretirar,v800,fine, wrist0\WObj:=wobj0;
                        abrir;
                        MoveL home,v800,fine, wrist0\WObj:=wobj0;
            ENDTEST
        ENDTEST
         CASE 5:
            STOP;
            DEFAULT:
            TPWrite "Elección ilegal";
    ENDTEST
ENDPROC

PROC abrir()

    waittime 1;
    reset doGripper;
    waittime 1;

ENDPROC


PROC cerrar()

    waittime 1;
    set doGripper;
    waittime 1;

ENDPROC
ENDMODULE
```

7 CONCLUSIONES

El objetivo inicial propuesto en el Proyecto Final de Carrera consistía en el desarrollo de diversas aplicaciones para dicho manipulador en lenguaje RAPID, la simulación mediante un software específico (Robotstudio) diseñado por ABB para sus modelos de robot y la realización de diversas guias de aprendizaje sobre el lenguaje RAPID y el software de simulación Robotstudio; superando el reto que supone aprender, comprender y poner en práctica todos los conocimientos nuevos necesarios para realizar este proyecto.

Se ha llegado a la conclusión de que es de gran importancia a la hora de realizar cualquier programa conocer los limites de configuración del robot ya que se precisa conocer las singularidades y los puntos críticos antes de llegar a ellos durante la ejecución de la aplicación, y por tanto es necesario un estudio previo de cada uno de los casos.

Además es de gran utilidad el software de RobotStudio proporcionado por ABB ya que permite recrear la estación sobre la cual se va a trabajar y emplearla como banco de pruebas para realizar diferentes aplicaciones sin tener que dañar el robot real, ni el entorno que lo rodea, y sobre todo para evitar provocar cualquier tipo de daño al usuario; permitiendo alcanzar la verificación de la operación completa del robot.

Por ello es conveniente que a pesar de las ventajas que conlleva la simulación offline en cualquier manipulador, siempre y cuando teniendo el software requerido, se ha de ser consciente de los diversos factores que influyen al robot real en su entorno para poder recrearlo rigurosamente, teniendo en cuenta sobre todo la desviación que pueda poseer el robot sobre todo si la aplicación tratada requiere un elevado grado de exactitud.

Como conclusión final se debería recalcar la gran versatilidad que proporciona este modelo de manipulador industrial que a pesar de su reducido tamaño y su limitado campo de trabajo debido a sus dimensiones es capaz de realizar una gran variedad de aplicaciones en la industrial con gran precisión.

Como futuras líneas de trabajo tras este proyecto se considerarán:

- Implementación de las practicas simuladas en robot real
- Comprobación de los resultados obtenidos con la ejecución de las practicas propuestas en esta memoria tanto en la simulación como en el robot real y verificar que son equivalentes.
- Sincronización de varios robots trabajando en una misma célula de trabajo
- Implementación de prácticas empleando visión artificial
- La comunicación entre Matlab y el robot para tareas de visión artificial

8 BIBLIOGRAFIA

[1] Antonio Barrientos, «Fundamentos de la robótica » *MCGraw*.

[2] https://robotapps.robotstudio.com/ revisado el 07/07/2017.

[3] http://reea-blog.blogspot.com.es/ revisado el 07/07/2017.

[4] http://tonirv1985.wixsite.com/automatizados revisado el 07/07/2017.

[5] https://es.scribd.com/document/67008043/Robot-Studio-3-1-2-Guia-Del-Usuario-Spanish revisado el 07/07/2017.

[6] http://www.abb.es/AbbLibrary/DownloadCenter/?showresultstab=true&browsecategory=9AAC910011&displayversion=1&dockind=Manual#&&/wEXAQUDa2V5BYMBM8KwOUFBQzkxMDAxMcKxwrHCsU1hbnVhbMKxwrFUwrHCsTTCsWVzwrExMDnCsUVTwrHCsTLCsVTCsTDCscKxMMKxMcKxMjDCsTIwwrExwrHCscKxwrHCsTlBQUM5MTAwMTHCscKxwrHCsTExwrHCscKxwrE2wrHCsTDCsUVTwrHCsTB6yK/7bPIdKY5C5sX08VpxxEdfSQ revisado el 07/07/2017.

[7] https://www.youtube.com/channel/UCMjjG2oBntG3j6Jtv5oxkDw revisado el 07/07/2017.

[8] https://www.youtube.com/channel/UCw9FCAj4waIm4KNmVHzlg3Q revisado el 07/07/2017.

[9] https://www.youtube.com/channel/UCsRQGtuqCm1-d-whRYOSV2Q revisado el 07/07/2017.

[10] https://www.youtube.com/watch?v=-T8YE9S8bHg revisado el 07/07/2017.

[11] https://library.e.abb.com/public/bedd1769ea1e4bb9c1257da10037e215/IRC5_IndustrialRobotController_ROB0295EN.pdf revisado el 07/07/2017.

[12] https://library.e.abb.com/public/2b5b950d68a0503cc1257c0c003cb703/3HAC041344-es.pdf revisado el 07/07/2017.

[13] http://www02.abb.com/global/essup/essup504.nsf/bf177942f19f4a98c1257148003b7a0a/f386f662bf8e8a16c125702a003f33dd/$file/irc5+2005+web.pdf revisado el 07/07/2017.

[14] https://library.e.abb.com/public/6aeb483836740e11c1257b4b0052375b/3HAC032104-005_revE_es.pdf revisado el 07/07/2017.

[15] https://library.e.abb.com/public/c13e1c5490c61230c125796000515137/IRC5%20datasheet%20PR10258%20EN_R13.pdf revisado el 07/07/2017.

[16] http://www.ferret.com.au/ODIN/PDF/Showcases/101058.pdf revisado el 07/07/2017.

ANEXO 1: CONCEPTOS

TOOL CENTER POINT (TCP): el punto central de la herramienta del robot es el punto cuyas coordenadas se almacenan en el programa. Se puede definir un TCP por cada herramienta que posea el robot. Normalmente consideraremos TCP0 el punto central de la herramienta cuando la muñeca no posea ninguna herramienta externa.

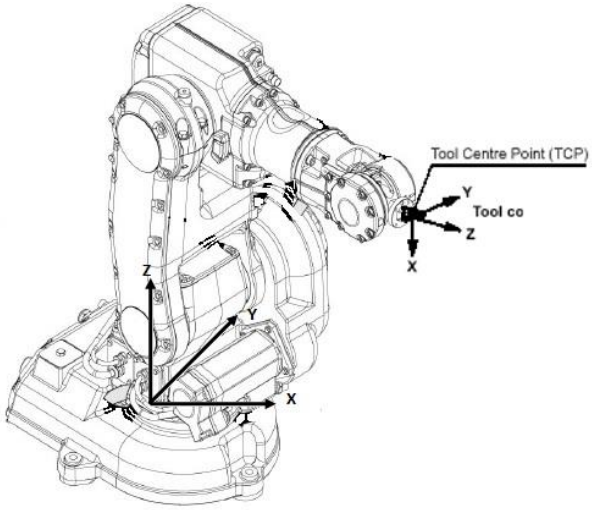


Figura 1. Tool Centre Point del robot IRB 120

RESOLUCIÓN: mínimo incremento que puede aceptar la unidad de control del robot. Su valor está limitado por:

- Resolución de los captadores de posición y los convertidores A/D y D/A.
- Número de bits con los que trabaja la CPU del robot.
- Elementos motrices (si son discretos. Por ejemplo: motor paso a paso)

PRECISIÓN: distancia entre el punto programado y el punto realmente alcanzado (valor medio tras varios ciclos). Su valor está limitado por errores de calibración del robot, deformaciones, errores de redondeo en el cálculo (puntos singulares) o por errores entre las dimensiones teóricas y reales del robot.

REPETIBILIDAD: es el radio de la esfera que comprende los puntos alcanzados por el robot tras realizar suficientes movimientos, al ordenarle ir al mismo punto de destino programado, con condiciones iguales de carga, temperatura, etc. Su valor está limitado por problemas en el sistema mecánico de transmisión (rozamientos, histéresis, zonas muertas)

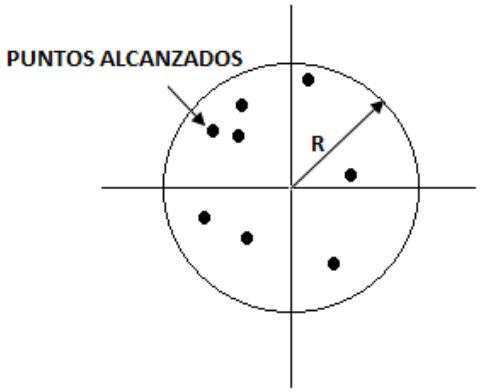


Figura 2. Zona de repetibilidad

ÁREA DE TRABAJO: es la región del espacio compuesta por todos los puntos que pueden ser alcanzados por el final del brazo del robot o por algún punto de su muñeca, pero sin considerar el elemento terminal.

Las especificaciones de los fabricantes de robots pueden excluir ciertas regiones del espacio de trabajo si en ellas el robot no pudiera alcanzar determinadas especificaciones de velocidad o carga. El volumen y la forma del espacio de trabajo son muy importantes a la hora de las aplicaciones industriales, ya que determinan las capacidades del robot.

GRADOS DE LIBERTAD (GDL): se considera grado de libertad a cada uno de los movimientos independientes que puede realizar una articulación de un robot respecto a la anterior. El número de grados de libertad determina la accesibilidad de un robot y su capacidad para orientar sus herramientas.

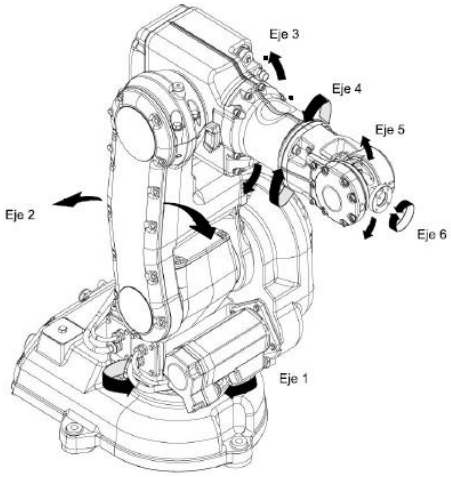


Figura 3. Robot ABB IRB 120 con seis grados de libertad

PUNTOS SINGULARES: puntos del espacio de trabajo del robot sobre los que no es posible realizar una trayectoria rectilínea. Implicaría el movimiento de algún eje a velocidad infinita, además el valor de los ejes en esa posición se encuentra indeterminado.

Matemáticamente podemos definirlo como el "problema inverso"; no tiene solución única, es decir, no existe la matriz jacobiana inversa, el determinante es nulo. Existen dos tipos de singularidades:

- Singularidades en los límites del espacio de trabajo: el robot no puede desplazarse en la dirección que lo aleje del espacio de trabajo.

- Singularidades en el interior del espacio de trabajo: se producen normalmente por el alineamiento de dos o más ejes.

CAPACIDAD DE CARGA: carga que es capaz de manipular el robot. Viene condicionada por el tamaño, la configuración y el sistema de accionamiento del robot. Se debe considerar el peso de la pinza más el de la pieza. Además de la carga (peso y c.d.g.) en algunos casos se deben también tener en cuenta los momentos de inercia.

VELOCIDAD: el dato proporcionado en los catálogos corresponde a la velocidad en régimen permanente (velocidad nominal), que es la que podría alcanzar el robot en largas distancias. En movimientos cortos son más significativos los tiempos de arranque y parada (aceleraciones y deceleraciones); es por ello que, en algunos robots, en lugar de la velocidad se indica el tiempo empleado en realizar un movimiento determinado, generalmente cuando se trata de robots dedicados a la manipulación rápida de piezas (pick and place).

Una velocidad de trabajo elevada aumenta el rendimiento del robot, pero la determinación de la velocidad adecuada depende también de la precisión que se desee en el posicionamiento, el peso a manipular y la distancia a recorrer.

ACELERACION: el objetivo en el movimiento de los robots es conseguir altas aceleraciones para alcanzar rápidamente la velocidad nominal programada. Para ello los fabricantes procuran minimizar las inercias, disminuyendo las masas de las partes que constituyen el brazo y acercando lo máximo posible sus centros de gravedad a los ejes de giro de cada articulación y al eje central del cuerpo del robot. En general, los robots más pequeños son más rápidos que los de gran envergadura.

ENTORNO DE TRABAJO: a menudo los robots sustituyen a los operarios humanos en tareas insalubres o potencialmente peligrosas. Deben operar en ambientes contaminados, bajos temperaturas extremas, en ambientes estériles…por lo que han de ser diseñados teniendo en cuenta estas restricciones.

ANEXO 2: DATASHEET

IRB 120:

Robotics

IRB 120
Industrial Robot
ABB's smallest robot – for flexible and compact production

The IRB 120 robot is the latest addition to ABB's new fourth-generation of robotic technology and ABB's smallest robot ever produced. Ideal for material handling and assembly applications, the new IRB 120 robot provides an agile, compact and lightweight solution with superior control and path accuracy.

Compact and lightweight
As the smallest robot from ABB, the IRB 120 offers all the functionality and expertise of the ABB range in a much smaller package. Its reduced weight of only 25kg and compact design enables it to be mounted virtually anywhere, whether it is inside a cell, on top of a machine or close to other robots on the production line.

Multipurpose
Ideal for a wide range of industries including the electronic, food and beverage, machinery, solar, pharmaceutical, medical and research sectors, the IRB 120 joins ABB's fourth-generation of new robotic technology.

A white finish Clean Room ISO class 5 version enhances this versatility by making it suitable for environment with stringent cleanliness standard.

The six-axis robot handles a payload of up to 3kg (4kg with its wrist down) and with a reach of 580 mm, the robot is able to carry out a series of operations using flexible rather than hard automated solutions. The IRB 120 is the perfect building block to design cost effective applications – especially when space is at a premium.

Easy to integrate
Weighing in at only 25kg, this robot arm is truly the most portable and easy to integrate on the market. It can be mounted at any angle without any restriction. The smooth surfaces are easy to clean and the cables for air and customer signals are internally routed, all the way from the foot to the wrist, ensuring that integration is effortless.

Optimized working range
In addition to a horizontal reach of 580 mm, the robot has best in class stroke and the ability to reach 112 mm below its base. Furthermore, the IRB 120 has a very compact turning radius, which is enabled by the robots symmetric architecture, without offset on axis 2. This ensures the robot can be mounted close to other equipment and the slim wrist enables the arm to reach closer to its application.

Fast, accurate and agile
Designed with a light, aluminum structure, the powerful compact motors ensure the robot is enabled with a fast acceleration, and can deliver accuracy and agility in any application. Using the IRB 120T variant, cycle-times can be reduced up to 25% where the work piece needs extensive re-orientation and axis 4, 5 and 6 are predominantly used. This faster versions is well suited for pick and packing applications and guided operations together with PickMaster 3™.

IRC5 Compact controller – optimised for small robots
ABB's new IRC5 Compact controller takes the capabilities of the extremely powerful IRC5 controller and presents them in a truly compact format. The new Compact controller brings accuracy and motion control to applications, which previously had been exclusive to large installations.

In addition to space saving benefits, the new controller also enables easy commissioning through one phase power input, external connectors for all signals and a built-in expandable 16 in, 16 out, I/O system.

RobotStudio for offline programming enables manufacturers to simulate a production cell to find the optimal position for the robot, and provide offline programming to prevent costly downtime and delays to production.

Reduced footprint
For applications where floor space is crucial, the combination of the new compact, lightweight architecture of the IRB 120 with the new IRC5 Compact controller introduces a significantly reduced footprint.

Power and productivity
for a better world™

IRB 120

Specification

Variants	Reach	Payload	Armload
IRB 120-3/0.6	580 mm	3 kg (4kg)*	0.3 kg

Features

Integrated signal supply	10 signals on wrist
Integrated air supply	4 air on wrist (5 bar)
Position repeatability	0.01 mm
Robot mounting	Any angle
Degree of protection	IP30
Controllers	IRC5 Compact / IRC5 Single cabinet

Movement

Axis movements	Working range	Maximum speed	
		IRB 120	IRB 120T
Axis 1 Rotation	+165° to -165°	250 °/s	250 °/s
Axis 2 Arm	+110° to -110°	250 °/s	250 °/s
Axis 3 Arm	+70° to -110°	250 °/s	250 °/s
Axis 4 Wrist	+160° to -160°	320 °/s	420 °/s
Axis 5 Bend	+120° to -120°	320 °/s	590 °/s
Axis 6 Turn	+400° to -400°	420 °/s	600 °/s

Performance

	IRB 120	IRB 120T
1 kg picking cycle		
25 x 300 x 25 mm	0.58 s	0.52 s
25 x 300 x 25 with 180° axis 6 reorientation	0.92 s	0.69 s
Acceleration time 0-1 m/s	0.07 s	0.07 s

Electrical connections

Supply voltage	200–600 V, 50/60 Hz
Rated power	
Transformer rating	3.0 kVA
Power consumption	0.25 kW

Physical

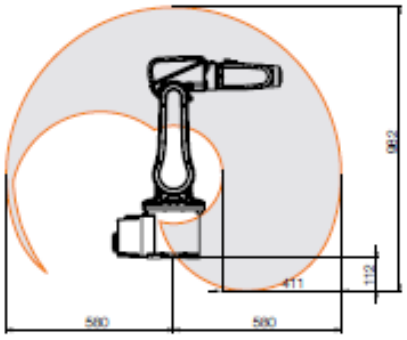
Dimension robot base	180 x 180 mm
Dimension robot height	700 mm
Weight	25 kg

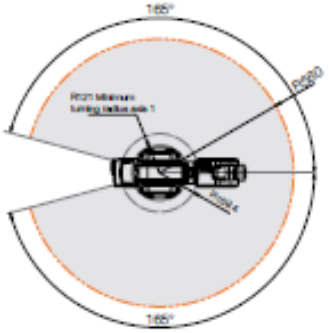
Environment

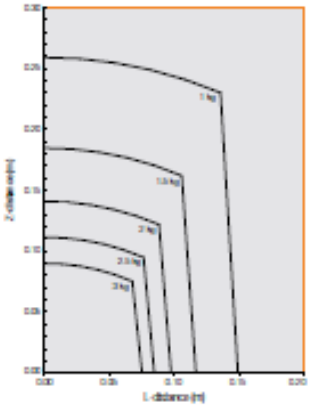
Ambient temperature for Robot manipulator:	
During operation	+5°C (41°F) to +45°C (122°F)
Relative transportation and storage	-25°C (-13°F) to +55°C (131°F)
For short periods	up to +70°C (158°F)
Relative humidity	Max 95%
Options	Clean Room ISO class 5 (certified by IPA)**
Noise level	Max 70 dB (A)
Safety	Safety and emergency stops 2-channel safety circuits supervision 3-position enabling device
Emission	EMC/EMI-shielded

* With vertical wrist
** ISO class 4 can be reached under certain conditions
Data and dimensions may be changed without notice

Working range at wrist center & load diagram







www.abb.com/robotics

Power and productivity for a better world™

IRC5 COMPACT CONTROLLER

Robotics

IRC5 Compact controller
Optimised for small robots

The IRC5 Compact controller extends the comprehensive IRC5 family of robot controllers. It brings the familiar benefits of the world leading robot controller, including superior motion control and flexible RAPID language, while adding the advantage of a minimised footprint.

IRC5 Compact
The IRC5 Compact offers the capabilities of the extremely powerful IRC5 controller in a truly compact format. In addition, the IRC5 Compact delivers space saving benefits and easy commissioning through one phase power input, external connectors for all signals and a built in expandable 16 in, 16 out I/O system.

Utilising many of the well-known features of the IRC5 controller, the compact version offers familiar programming and operation, ensuring no additional training is required.

The IRC5 Compact controller is available for the lower end robots of the IRB range.

Safety
Operator safety is a central quality of the IRC5 Compact, fulfilling all relevant regulations, as certified by third-party inspections.

Motion Control
Based on advanced dynamic modeling, the IRC5 optimizes the performance of the robot for the shortest possible cycle time (QuickMove) and precise path accuracy (TrueMove). Together with a speed independent path, predictable and high performance behaviour is delivered automatically, with no tuning required by the programmer. What you program is what you get!

FlexPendant
The FlexPendant is characterised by its clean, colour touch screen-based design and 3D joystick for intuitive interaction. Powerful customized application support enables loading of tailor-made applications, for example operator screens, thus eliminating the need for a separate operator HMI.

RAPID programming language
RAPID programming provides the perfect combination of simplicity, flexibility and power. It is a truly unlimited language with support for structured programs, shop floor language and advanced features. It also incorporates powerful support for many process applications.

Communication
The IRC5 supports the state-of-the-art field busses for I/O and is a well-behaved node in any plant network. Sensor interface functionality, remote disk access and socket messaging are examples of the many powerful networking features.

Remote service enabled
Remote monitoring of the robot is available through standard communication networks (GSM or Ethernet). Advanced diagnostic methods allow fast investigation on failure as well as monitoring of the robot condition throughout the life cycle. Service packages are available, including new services like backup management, reporting and proactive maintenance activities.

RobotStudio
RobotStudio is a powerful PC tool for working with IRC5 data. It can be used offline, providing a perfect digital copy of the automation system together with strong programming and simulation features.

Power and productivity for a better world™

IRC5 Compact

Specification	
Controller hardware:	Multi-processor system
	PCI bus
	Flash disk mass memory
	Energy back-up power failure handling
	USB memory interface
Control software:	Well proven real-time OS
	High-level RAPID programming language
	PC-DOS file format Preloaded software,
	available on DVD
	Extensive functionality set, see separate
	RobotWare data sheet

Electrical Connections	
Supply voltage:	Single phase 220/230 V, 50-60 Hz

Physical		
	Size HxWxD	**Weight**
	258 x 450 x 580	28.5 kg

Environment	
Ambient temperature:	+ 0° C (32°F) - +45°C (113°F)
Relative humidity	Max. 95%
Level of protection	IP20
Fulfilment of regulations	Machine directive 98/37/EC regulation
	Annex II B
	EN 60204-1:2006
	ISO 10218-1:2006
	ANSI/RIA R 15.06 -1999

User Interfaces	
Control panel	On cabinet or remote
FlexPendant	Weight 1 kg
	Graphical color touch screen
	Joystick
	Emergency stop
	Support for right- and left hand operators
	USB Memory support
Maintenance	Diagnostic software
	Recovery procedure
	Logging with time stamp
	Remote Service enabled
Supported robots	IRB 120
	IRB 140
	IRB 260
	IRB 360
	IRB 1410
	IRB 1600

Safety	
Basic:	Safety and emergency stops
	2-channel safety circuits supervision
	3-position enabling device

Machine Interfaces	
Inputs/outputs:	Standard 16/16 (up to 2200)
Digital:	24V DC or relay signals
Analogue:	2 x 0-10V, 3x ±10V, 1x4-20mA
Serial channel:	1 x RS 232 (RS422 with adaptor)
Network:	Ethernet (10/100 Mbits per second)
Two channels:	Service and LAN
Fieldbus Master:	DeviceNet™
	PROFIBUS DP
	Ethernet/IP™
Fieldbus Slave:	PROFINET
	PROFIBUS DP
	Ethernet/IP™
	Allen-Bradley Remote I/O
	CC-link
Conveyor encoder	Up to 6 channels

Sensor Interfaces	
	Search stop with automatic program shift
	Seam/contour tracking
	Conveyor tracking
	Machine vision
	Force control

Data and dimensions may be changed without notice

ANEXO 3: COMANDOS ROBOTSTUDIO

Navegador de Diseño

Icono	Nodo	Descripción
 xx050000	Robot	El robot en la estación.
 xx050001	Herramienta	Una herramienta.
 xx050002	Colección de enlaces	Contiene todos los enlaces de los objetos.
 xx050003	Eslabón	Un objeto físico en una conexión de articulación. Cada eslabón está formado por una o varias piezas.
 xx050004	Bases de coordenadas	Contiene todas las bases de coordenadas de un objeto.
 xx050005	Grupo de componentes	Una agrupación de piezas u otros conjuntos que tienen sus propios sistemas de coordenadas. Se utiliza para estructurar una estación.
 xx050006	Componente	Un objeto físico de RobotStudio. Las piezas basadas en información geométrica se construyen a partir de una o varias entidades bidimensionales o tridimensionales. Las piezas que no contienen información geométrica (por ejemplo los archivos .jt importados) están vacías.
 xx050007	Conjunto de colisión	Contiene todos los conjuntos de colisión. Cada conjunto de colisión incluye dos grupos de objetos.
 xx050008	Grupo de objetos	Contiene referencias a los objetos que están sujetos a la detección de colisiones.
 xx050009	Mecanismos de conjunto de colisión	Los objetos del conjunto de colisión.
 xx050010	Base de coordenadas	Las bases de coordenadas de la estación.

Navegador trayectorias y objetivos

Icono	Nodo	Descripción
 xx050011	**Estación**	Su estación en RobotStudio.
 xx050012	**Controlador virtual**	Es el sistema utilizado para controlar los robots, al igual que un controlador IRC5 real.
 xx050013	**Tarea**	Contiene todos los elementos lógicos de la estación, como objetivos, trayectorias, objetos de trabajo, datos de herramienta instrucciones.
 xx0500001376	**Colección de datos de herramienta**	Contiene todos los datos de herramienta.
 xx050014	**Datos de herramienta**	Un dato de herramienta para un robot o una tarea.
 xx050015	**Objetos de trabajo y objetivos**	Contiene todos los objetos de trabajo y objetivos de la tarea o del robot.
 xx050016	**Colección de objetivos de ejes y objetivo de ejes**	Una posición especificada de los ejes del robot.
 xx050017	**Colección de objetos de trabajo y objeto de trabajo**	El nodo de colecciones de objetos de trabajo y los objetos de trabajo que contiene.
 xx050018	**Objetivo**	Una posición y una rotación definidas para el robot. Un objetivo equivale a un RobTarget en un programa de RAPID.
 xx050019	**Objetivo sin configuración asignada**	Un objetivo que no tiene ninguna configuración de ejes asignada, por ejemplo un objetivo reposicionado o un nuevo objetivo creado por un medio distinto de la programación.
 xx050020	**Objetivo sin configuración encontrada**	Un objetivo inalcanzable, es decir, para el que no se ha encontrado ninguna configuración de ejes.
 xx050021	**Colección de trayectorias**	Contiene todas las trayectorias de la estación.
 xx050022	**Trayectoria**	Contiene instrucciones para los movimientos del robot.

Icono	Nodo	Descripción
 xx050023	**Instrucción de movimiento lineal**	Un movimiento lineal del TCP hacia un objetivo. Si el objetivo no tiene ninguna configuración asignada, la instrucción de movimiento recibe los mismos símbolos de aviso que el objetivo.
 xx050024	**Instrucción de movimiento de ejes**	Un movimiento de ejes hacia un objetivo. Si el objetivo no tiene ninguna configuración asignada, la instrucción de movimiento recibe los mismos símbolos de aviso que el objetivo.
 xx050025	**Instrucción de acción**	Define una acción que debe ser realizada por el robot en una ubicación determinada de una trayectoria.

Navegador de Modelado

Icono	Nodo	Descripción
 modeling	**Componente**	Elementos geométricos que corresponden a los objetos del navegador **Diseño**.
 modelin0	**Cuerpo**	Elementos geométricos básicos a partir de los cuales se componen las piezas. Los cuerpos en 3D contienen varias caras, los cuerpos en 2D contienen una cara y curvas que no tienen ninguna cara.
 modelin1	**Cara**	Las caras de los cuerpos.

Navegador Controlador

Icono	Nodo	Descripción
 controll	**Controladores**	Contiene los controladores que están conectados a la vista de robot.
 control0	**Controlador conectado**	Representa un controlador que tiene una conexión en funcionamiento.
 control1	**Controlador en conexión**	Representa un controlador que se está conectando en este momento.
 control2	**Controlador desconectado**	Representa un controlador que ha perdido su conexión. Puede deberse a que ha sido apagado o desconectado de la red.
 control3	**Inicio de sesión denegado**	Representa un controlador que deniega el acceso para el inicio de sesión. Las causas posibles para la denegación del acceso son: • El usuario carece de los privilegios necesarios • Hay demasiados clientes conectados al controlador. • La versión de RobotWare del sistema que se está ejecutando en el controlador es más reciente que la versión de RobotStudio
 configu0	**Configuración**	Contiene los temas de configuración.
 configu1	**Tema**	Cada tema de parámetro se representa con un nodo: • Comunicación • Controlador • I/O • Comunicación hombre-máquina • Movimiento
 eventrec	**Registro de eventos**	Con el Registro de eventos puede ver y guardar los eventos del controlador.
 io	**Sistema de E/S**	Representa el sistema de E/S del controlador. El sistema de E/S se compone de redes industriales y unidades.

 io-node	**Red Industrial**	Una red industrial es un conector para uno o varios dispositivos.
 io-devic	**Dispositivo**	Un dispositivo es una tarjeta, un panel o cualquier otro dispositivo dotado de puertos a través de los cuales se envían las señales de E/S.
 rapid16t	**Tareas de RAPID**	Contiene las tareas activas del controlador (programas).
 prgintas	**Tarea**	Una tarea es un programa de robot que se ejecuta de forma independiente o junto con otros programas. Un programa se compone de un conjunto de módulos.
 xx1500000335	**Módulos de programa**	Los módulos de programa contienen un conjunto de declaraciones de datos y rutinas para una tarea determinada. Los módulos de programa contienen datos específicos de esta tarea.
Módulos de sistema	**Módulos de sistema**	Los módulos de sistema contienen un conjunto de declaraciones de tipo, declaraciones de datos y rutinas. Los módulos de sistema contienen datos que se aplican al sistema de robot, independientemente de qué módulos de programa estén cargados.
 nostepin	**Módulo NOSTEPIN**	Un módulo en el que no se puede entrar detalladamente durante la ejecución paso a paso. Es decir, todas las instrucciones del módulo se tratan como un solo programa si éste se ejecuta en el modo paso a paso.
 modules	**Módulos de programa de sólo visualización y sólo lectura**	Un icono para los módulos de programa que son de solo visualización o solo lectura.
 module_e	**Módulos de sistema de sólo visualización y sólo lectura**	Un icono para los módulos de sistema que son de solo visualización o solo lectura.
 procedur	**Procedimiento**	Una rutina que no devuelve ningún valor. Los procedimientos se utilizan como subprogramas.
 function	**Función**	Una rutina que devuelve un valor de un tipo específico.
 trap16tr	**Rutina TRAP**	Una rutina que proporciona una forma de responder a las interrupciones.

Navegador Archivos

Icono	Nodo	Descripción
 xx1200000824	**Archivos**	Consulte *Administración de archivos de RAPID en la página 483*.
 xx1200000825	**Copias de seguridad**	Consulte *Administración de copias de seguridad del sistema en la página 483*.

Navegador complementos

Icono	Nodo	Descripción
 xx1200000826	**Complemento**	Indica un complemento disponible cargado en el sistema
 xx1200000827	**Complemento desactivado**	Indica un complemento desactivado
 xx1200000828	**Complemento descargado**	Indica un complemento descargado del sistema

Navegación por la ventana de gráficos con ayuda del ratón

Para	Use la combinación de teclado/ratón	Descripción
Seleccionar elementos  selectio	 left-cli	Simplemente haga clic en el elemento a seleccionar. Para seleccionar más de un elemento, mantenga presionada la tecla CTRL mientras hace clic en los nuevos elementos.
Girar la estación  rotate	**CTRL** + **MAYÚS** +  left-cli	Presione CTRL + MAYÚS + botón izquierdo del ratón mientras arrastra el ratón para girar la estación. Con un mouse de 3 botones, puede usar los botones central y derecho en lugar de la combinación de teclado.

Desplazar manualmente la estación  pan	**CTRL** +  left-cli	Presione CTRL + botón izquierdo del ratón mientras arrastra el ratón para desplazar manualmente la estación.
Aplicar o reducir la estación  zoom	**CTRL** +  right-cl	Presione CTRL + botón derecho del ratón mientras arrastra el ratón hacia la izquierda para reducir. Arrastre hacia la derecha para ampliar. Con un mouse de 3 botones, también puede usar el botón central en lugar de la combinación de teclado.
Ampliar o reducir con una ventana  window_z	**MAYÚS** +  right-cl	Presione MAYÚS + botón derecho del ratón mientras arrastra el ratón a través del área que desea ampliar.
Seleccionar con una ventana  window_s	**MAYÚS** +  left-cli	Presione MAYÚS + botón izquierdo del ratón mientras arrastra el ratón a través del área para seleccionar todos los elementos que correspondan al nivel de selección actual.

Selección de un elemento en la ventana de gráficos

Para seleccionar elementos en la ventana de gráficos, realice las operaciones siguientes:

1. En la parte superior de la ventana Gráfico, haga clic en el icono del nivel de selección deseado.
2. Opcionalmente, haga clic en el modo de ajuste deseado para la parte del elemento que desee seleccionar.
3. En la ventana de gráficos, haga clic en el elemento. El elemento seleccionado se resaltará.

Selección múltiple de elementos en la ventana de gráficos:

- Presione la tecla **MAYÚS** y, en la ventana de gráficos, arrastre el ratón diagonalmente sobre los objetos a seleccionar.
- Para seleccionar elementos separados: En el navegador, mantenga presionada la tecla **CTRL** y haga clic en los elementos que desee seleccionar. Los elementos se resaltarán.

Métodos abreviados de teclado generales

Comando	Combinación de teclas
Métodos abreviados generales	
Activar la barra de menús	F10
Abrir la Ayuda de la API	ALT + F1
Abrir la Ayuda	F1
Abrir Virtual FlexPendant	CTRL + F5
Cambiar de una ventana a otra	CTRL + TAB
Comandos generales	
Añadir sistema de controlador	F4
Abrir estación	CTRL + O
Hacer una captura de pantalla	CTRL + B
Programar instrucción	CTRL + MAYÚS + R
Programar un posición	CTRL + R
Importar geometría	CTRL + G
Importar biblioteca	CTRL + J
Nueva estación vacía	CTRL + N
Guardar estación	CTRL + S
Comandos generales de edición	
Copiar	CTRL + C
Cortar	CTRL + X
Pegar	CTRL + V
Eliminar	SUPR
Rehacer	CTRL + Y
Actualización	F5
Cambiar nombre	F2
Seleccionar todo	CTRL + A
Guardar todo	Ctrl+Mayús+S
Deshacer	CTRL + Z

Métodos abreviados de teclado editor rapid

Comando	Combinación de teclas
Intellisense del Editor de RAPID	
Palabra completa	CTRL + ESPACIO
Información de parámetros	CTRL + MAYÚS + ESPACIO
Autocompletar	TAB (cuando el cursor está situado al final de un identificador)
Comandos generales del Editor de RAPID	
Iniciar ejecución de programa	F8
Paso a paso por instrucciones	F11
Paso a paso para salir	MAYÚS + F11
Paso a paso por procedimientos	F12
Parar	MAYÚS + F8
Activar/desactivar punto de interrupción	F9
Aplicar cambios	CTRL + MAYÚS + S
Imprimir	CTRL + P
Comandos de texto del Editor de RAPID	
Copiar	CTRL + Insert o bien CTRL + C
Cortar	MAYÚS + Supr o bien CTRL + X
Cortar línea	CTRL + L
Eliminar línea	CTRL + MAYÚS + L
Borrar el principio de la palabra	CTRL + RETROCESO
Borrar el final de la palabra	CTRL + Supr
Buscar la siguiente aparición	F3
Aumentar margen	Pestaña
Cambiar a minúsculas el texto seleccionado	CTRL + U
Cambiar a mayúsculas el texto seleccionado	CTRL + MAYÚS + U

Comando	Combinación de teclas
Ir al principio del documento	CTRL + Inicio
Ir al principio de la línea	Inicio
Ir al final del documento	CTRL + Fin
Ir al final de la línea	Final
Ir a la siguiente palabra	CTRL + Derecha
Ir a la palabra anterior	CTRL + Izquierda
Ir a la parte inferior visible	CTRL + Av Pág
Ir a la parte superior visible	CTRL + Re Pág
Minimizar la cinta	Ctrl+F1
Abrir la línea superior	CTRL + Intro
Abrir la línea inferior	CTRL + MAYÚS + Intro
Reducir margen	MAYÚS + Tabulador
Pegar	MAYÚS + Insert o bien CTRL + V
Rehacer	CTRL + MAYÚS + Z o bien CTRL + Y
Desplazar hacia abajo	CTRL + Abajo
Desplazar hacia arriba	CTRL + Arriba
Seleccionar el bloque inferior	ALT + MAYÚS + Abajo
Seleccionar el bloque de la izquierda	ALT + MAYÚS + Izquierda
Seleccionar el bloque de la derecha	ALT + MAYÚS + Derecha
Seleccionar el bloque superior	ALT + MAYÚS + Arriba
Seleccionar hacia abajo	MAYÚS + Abajo
Seleccionar hacia la izquierda	MAYÚS + Izquierda
Seleccionar una página hacia abajo	MAYÚS + Av Pág
Seleccionar una página hacia arriba	MAYÚS + Re Pág
Seleccionar hacia la derecha	MAYÚS + Derecha
Seleccionar hasta el principio del documento	CTRL + MAYÚS + Inicio
Seleccionar hasta el principio de la línea	MAYÚS + Inicio
Seleccionar hasta el principio del documento	CTRL + MAYÚS + Fin
Seleccionar hasta el fin de la línea	MAYÚS + Fin
Seleccionar hasta la siguiente palabra	CTRL + MAYÚS + Derecha
Seleccionar hasta la palabra anterior	CTRL + MAYÚS + Izquierda
Seleccionar hasta la parte inferior visible	CTRL + MAYÚS + Av Pág
Seleccionar hasta la parte superior visible	CTRL + MAYÚS + Re Pág
Seleccionar hacia arriba	MAYÚS + Arriba
Seleccionar una palabra	CTRL + MAYÚS + W
Captura de pantalla	Ctrl+B

Comando	Combinación de teclas
Situar puntero de programa en Main en todas las tareas	Ctrl+Mayús+M
Activar/desactivar punto de interrupción	F9
Activar/desactivar el modo de sobrescritura	Insert
Transponer caracteres	CTRL + T
Transponer líneas	CTRL + ALT + MAYÚS + T
Transponer palabra	CTRL + MAYÚS + T
Virtual FlexPendant	Ctrl+F5